MODELADO UML DEL GENERADOR DE CÓDIGO DE APLICACIONES WEB TGENP

Adárraga Mejía Carlos Alberto
Oliveros Villanueva Carlos Andrés

Trabajo de Investigación presentado para optar al título de ESPECIALISTA EN DESARROLLO SOFTWARE

Director
Ing. Luis Garrido
Docente del programa de Ingeniería de Sistemas

UNIVERSIDAD DEL MAGDALENA
FACULTAD DE INGENIERÍA
PROGRAMA DE ESP. DESARROLLO SOFTWARE
SANTA MARTA. D.T.C.H.
2014

MODELADO UML DEL GENERADOR DE CÓDIGO DE APLICACIONES WEB TGENP

Adárraga Mejía Carlos Alberto
Oliveros Villanueva Carlos Andrés

Trabajo de Investigación presentado para optar al título de ESPECIALISTA EN DESARROLLO SOFTWARE

Director
Ing. Luis Garrido
Docente del programa de Ingeniería de Sistemas

UNIVERSIDAD DEL MAGDALENA
FACULTAD DE INGENIERÍA
PROGRAMA DE ESP. DESARROLLO SOFTWARE
SANTA MARTA. D.T.C.H.

Tabla de Contenido

1. INFORMACIÓN GENERAL DEL PROYECTO: 4

2. RESUMEN DEL PROYECTO 5

3. DESCRIPCIÓN DEL PROYECTO 6

3.1 Planteamiento de la pregunta problema o de investigación y su justificación en términos de necesidades. 6

3.1.1 Justificación 7

3.2 Marco Teórico 8

3.3 Objetivos 16

3.3.1 General 16

3.3.2 Específicos 16

3.4. Metodología 16

3.4.1. Definición y delimitación del componente de software. 17

3.4.2. Recolección de funcionalidades existentes. 17

3.4.3. Generación automática de los diagramas de clase y de paquetes mediante herramientas CASE. 17

3.4.4. Análisis de código fuente. 17

3.5 Resultados/Productos Esperados y Potenciales Beneficiarios 18

3.6 Impactos esperados a partir del uso de los resultados: 18

3.7 Cronograma de actividades 19

4. PRESUPUESTO 19

5. RESULTADOS 20

REFERENCIAS 47

Tabla de Ilustraciones

Ilustración 1. Diagrama de Clases 10

Ilustración 2. Diagrama de Actividades; Decisiones 11

Ilustración 3. Diagrama de Actividades; Concurrencia 11

Ilustración 4. Diagrama de Secuencias 12

Ilustración 5. Tipos de Mensajes del Diagrama de Secuencias 12

Ilustración 6. Proceso de Ingeniería Inversa 13

Ilustración 7. Diagrama de Gantt del Cronograma de Actividades 19

Ilustración 8. Diagrama de Casos de Uso 21

Ilustración 9. Conexión a Base de Datos 31

Ilustración 10. Bienvenida TGENP 31

Ilustración 11. Validar datos conexión 31

Ilustración 12. Error 1 conexión a BD 32

Ilustración 13. Error 2 conexión a BD 32

Ilustración 14. Selección de Tablas a crear 32

Ilustración 15. Generación de Proyecto 33

Ilustración 16. Resultado de Generación de Proyecto Completo 33

Ilustración 17. Mensaje tabla sin campos 33

Ilustración 18. Lista de tablas para generar controladores 34

Ilustración 19. Mensajes con controladores creados 34

Ilustración 20. Mensaje "Tabla no posee campos" para generación de controladores 34

Ilustración 21. Borrar todo 35

Ilustración 22. Error Borrar Todo 35

Ilustración 23. Error Salir 35
Ilustración 24. Diagrama de Clases 40
Ilustración 25. Diagrama de paquetes 41
Ilustración 26. Diagrama de Actividad Conectar a Base Datos 43
Ilustración 27. Diagrama de Actividad Generar Proyecto 44
Ilustración 28. Diagrama de Secuencia Conectar Con Base de Datos 44
Ilustración 29. Diagrama de secuencias Generar proyecto Completo 46
Ilustración 30. Diagrama de Componentes 46

1. INFORMACIÓN GENERAL DEL PROYECTO:

	UNIVERSIDAD DEL MAGDALENA **FACULTAD DE INGENIERÍA** **FORMATO DE PRESENTACION DE PROPUESTAS INICIALES PARA LA ESPECIALIZACIÓN EN DESARROLLO DE SOFTWARE**	**Página: 1 de 6**

1. DATOS GENERALES DEL PROYECTO DE INVESTIGACIÓN.		
Nombre del Proyecto de Investigación: MODELADO UML DEL GENERADOR DE CODIGO DE APLICACIONES WEB TGENP		
Nombre del Grupo de Investigación*:		
Línea de Investigación: Desarrollo de Software		
Nombre de Estudiantes (Máximo 2): Ing. Carlos Oliveros, Ing. Carlos Adárraga		
Nombre del Director: Ing. Luis Garrido		
Total de Participantes (Estudiantes + Director de Trabajo de Grado): 3		
Duración del Proyecto (En Meses): 2	**Año de Financiación:**	
Correos Electrónicos: carlosadarragamejia@gmail.com, ing.oliveros87@gmail.com, luis.garrido.b@gmail.com		
Lugar de Ejecución del Proyecto:		
Tipo de Proyecto:		
Tipo de Entidad:		
Sede de la Entidad:		
Investigación Básica: X	**Investigación Aplicada:___**	**Desarrollo Tecnológico o Experimental: ___**
Capacidad Instalada: 3.550.000	Efectivo: 4.440.000	
Valor total del Proyecto: 7.990.000		
Descriptores / Palabras claves: UML, Generador de código, Framework, Ingeniería Inversa		

3 investigadores expertos locales, nacionales e internacionales en el tema de su propuesta y que estén en capacidad de evaluar proyectos en esta temática. Identificar nombres completos, direcciones electrónicas y decanaturas o centros de investigación al que pertenecen. (No significa que necesariamente los nombres señalados sean los que evalúen este proyecto en particular):

Nombres	Correo Electrónico	Centro de Inv.
	-4-	

2. RESUMEN DEL PROYECTO

En la actualidad los proyectos de software son cada vez más ambiciosos y complejos en cuanto a sus alcances, tiempos de ejecución y entrega, lo cual ha obligado a que los profesionales de la industria del software se apoyen en herramientas que les permitan agilizar la escritura del código fuente de las aplicaciones que desarrollan, herramientas conocidas como *generadores de código o FRAMEWORKS*.

Existen muchos generadores de código gratuitos y pagos en el mercado que funcionan para diversos lenguajes de programación, ambientes y motores de bases de datos. Cada uno de ellos con un nivel de dificultad de uso diferente pero al mismo tiempo con un nivel de robustez y alcance frente al producto final deseado.

En este sentido, ya no es suficiente para un desarrollador de software conocer una tecnología o lenguaje de programación especifico, pues para ser más competitivo en el mercado actual los profesionales de software deben estar familiarizados con el concepto y utilización de generadores de código o frameworks.

Desarrollar un generador de código propio como herramienta personal para el desarrollo de proyectos de software le permite al autor personalizar, adaptar y mejorar continuamente el mismo frente a las necesidades que puedan surgir durante el quehacer de su oficio sin depender de terceros. Pero como en todo proyecto de software este proceso requiere de una fase de diseño y modelado del sistema.

El presente proyecto pretende realizar el modelado UML (Unified Modeling Lenguage) de la segunda versión del Generador de código de aplicaciones WEB ***TGENP***, desarrollado por el Ing. Carlos Oliveros. TGENP es una aplicación web desarrollada con el lenguaje de programación PHP con la arquitectura de desarrollo MVC.

La principal funcionalidad de TGENP es la generar todo el código fuente PHP con la arquitectura de desarrollo MVC de una base de datos MYSQL dada, para tener en cuestión de minutos un primer prototipo funcional de la aplicación.

Entre las acciones de este primer prototipo funcional de la aplicación están:

- Creación, edición, consulta básica, consulta avanzada y consultas anidadas de datos.
- Exportación de listas a Excel y PDF.

- Exportación a PDF de vistas detalladas.
- Impresión de listas y vistas detalladas.
- Creación automática del módulo de Gestión de Usuarios, donde se pueden administrar los usuarios, permisos y estructuras de navegación (menús) de roles de usuarios, sistemas y manuales de los mismos. Además de toda la gestión de sesiones de usuarios.
- Funcionalidades JavaScript usando el Framework JQUERY, como creación de validaciones de datos para los formularios con Jquery.Validate, además de otras funcionalidades como autocompletados, organización de tablas, envió de datos por AJAX, entre otras funcionalidades usando otros plugins del mismo Framework (JQUERY).
- Combinación de formularios.
- Creación de todo el ORM en los modelos de la aplicación.
- Gestión de los mensajes para los usuarios de la aplicación que debe recibir después de realizar una acción; mensajes de éxito, error, informativo y de advertencia.

3. DESCRIPCIÓN DEL PROYECTO

3.1 Planteamiento de la pregunta problema o de investigación y su justificación en términos de necesidades.

Los profesionales del Software a medida que ejercen su oficio van adoptando patrones de diseño de código fuente (como MVC) y con el tiempo se hace evidente que algunos comportamientos y/o funcionalidades de estos diseños son comunes en la mayoría de los proyectos que desarrollan; como consultar, crear, actualizar y eliminar registros (CRUD), formularios, listas, control de sesiones de usuarios, permisos de perfiles, etc. Debido a lo anterior nace la idea de desarrollar el generador de código TGENP, que a experiencia del autor (ingeniero Carlos Oliveros) considera comunes en la mayoría de los sistemas de información.

Sin embargo la falta de documentación pertinente le dificulta a los desarrolladores diferentes al autor entender la estructura y funcionamiento del generador y del código fuente de las aplicaciones generadas. Incluso el mismo autor en ocasiones padece estas mismas dificultades debido a la complejidad que alberga una herramienta de esta magnitud. Otro punto importante es la necesidad de

evolucionar el generador a una nueva versión que permita una mejor distribución del tiempo para los desarrolladores y analistas que utilicen esta herramienta.

Atendiendo lo expuesto anteriormente, ¿qué hacer para que el funcionamiento y la estructura de TGENP pueda ser entendida más rápida y fácilmente por otros desarrolladores para que puedan usarlo en los diferentes proyectos de software donde se requiera y/o tengan la oportunidad de aportar en la mejora continua del generador hasta llegar a una segunda versión?.

3.1.1 Justificación

"Hoy en día, UML está consolidado como el lenguaje estándar en el análisis y diseño de sistemas de cómputo. Mediante UML es posible establecer la serie de requerimientos y estructuras necesarias para plasmar un sistema de software previo al proceso intensivo de escribir código.

En otros términos, así como en la construcción de un edificio se realizan planos previo a su construcción, en Software se deben realizar diseños en UML previa codificación de un sistema, ahora bien, aunque UML es un lenguaje, éste posee más características visuales que programáticas, mismas que facilitan a integrantes de un equipo multidisciplinario participar e intercomunicarse fácilmente, estos integrantes siendo los analistas, diseñadores, especialistas de área y desde luego los programadores." (Osmosis Latina, 2007, párr. 1 - 2, http://www.osmosislatina.com/lenguajes/uml/basico.htm)

Al pretender comenzar el desarrollo de una nueva versión de TGENP, al contrario de la primera versión se busca que esta vez se haga desde la participación de un equipo de trabajo comprometido con cada una de las fases de desarrollo y tareas que involucra un proyecto de software aplicando los estándares de calidad en el Desarrollo de Software, lo que hace relevante implementar un lenguaje que permita la participación e intercomunicación de forma fácil entre los participantes, más aún cuando se parte desde un desarrollo previo sin ningún tipo de documentación.

Es en este punto donde UML se convierte en una potente herramienta que permite la participación e intercomunicación de forma fácil entre los participantes, brindando un entendimiento certero y fácil del funcionamiento y estructura del generador de código en su versión actual, permitiendo que herramientas de calidad en el desarrollo de aplicaciones puedan ayudar a estandarizar y documentar estos procesos de análisis, diseño y desarrollo de la nueva versión de TGENP.

3.2 Marco Teórico

Se podría definir a un generador de código como una *herramienta de software que genera código fuente de software bajo una arquitectura predefinida.* Esta teoría o concepto puede variar en cada desarrollador, al igual que la definición de *Framework*.

Según el artículo *"Framework de desarrollo: un método ágil para el desarrollo de software"* de Eduardo Morcillo [1], se define a un Framework como "una estructura conceptual y tecnológica, formada por un conjunto de bloques predefinidos de software, cuya utilización permite la organización y el desarrollo de proyectos software de forma mucho más ágil".

Los frameworks y generadores de códigos comúnmente utilizados por los desarrolladores de software profesionales usan la arquitectura de desarrollo MVC (Modelo, Vista, Controlador), sobre todo en aplicaciones orientadas a WEB (Aplicaciones que se utilizan mediante un navegador WEB accediendo a una dirección específica) con lenguajes de programación como PHP, JSP, ASP.NET, entre otros, siendo estos los más populares.

EL patrón o arquitectura MVC separa las aplicaciones en 3 componentes principales:

- Modelo: Gestiona toda la parte de acceso a datos de una aplicación, como por ejemplo a una Base de Datos.
- Vista: Conformada por las interfaces de cara al usuario, como por ejemplo los formularios de creación yo edición, listas, etc.
- Controlador: Recibe las peticiones o eventos que vienen de la Vista activadas por el usuario e invoca a los Modelos para dar respuesta a estas peticiones. Procesa toda la información obtenida de los modelos, y se la pasa a las Vistas para que estas se la presenten de una forma más adecuada al usuario. En pocas palabras, es donde se codifica toda la lógica de negocio de la aplicación.

Las aplicaciones con arquitectura de desarrollo MVC pueden ser modeladas muy fácilmente con UML o Lenguaje Unificado de Modelado. UML es como su nombre lo indica, un lenguaje de modelado que tiene cuatro objetivos concretos que describe de la siguiente forma el autor *Enrique Hernández* en el artículo *"Lenguaje Unificado de modelado (UML)"*:

- *Visualizar: UML permite expresar de una forma gráfica un sistema de forma que otro lo puede entender [2].*
- *Especificar: UML permite especificar cuáles son las características de un sistema antes de su construcción [2].*
- *Construir: A partir de los modelos especificados se pueden construir los sistemas diseñados [2].*
- *Documentar: Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión [2].*

Los objetivos de UML dejan ver lo importante que es el modelado de cualquier aplicación WEB antes de comenzar la fase de desarrollo como tal, y es por esto que todas las grandes empresas de informática en el mundo además de que todas las herramientas CASE de desarrollo adoptaron UML como lenguaje de modelado.

Adentrándonos más en lo que es UML se puede evidenciar que su implementación como lenguaje de modelado ante un proyecto software permitirá desde la fase de análisis y diseño del proyecto alcanzar claridad y definir a precisión aspectos claves ante un nuevo desarrollo software como lo son su estructura, comportamiento e iteración, aspectos evidenciables por ejemplo en diagramas UML denominados diagramas de clases, diagrama de actividades y diagrama de secuencias respectivamente.

Una Clase es una categoría o grupo de cosas que tienen atributos y acciones similares. En la representación de un *Diagrama de Clases* se puede observar un rectángulo subdividido en tres celdas horizontales, las cuales en la primera celda muestra el nombre de la clase, en la segunda los atributos y en una tercera los métodos o acciones.

Es importante resaltar que las clases se relacionan entre sí, y para dichas relaciones existen elementos gráficos que indican el tipo de relación existente entre dos clases (Ilustración 1):

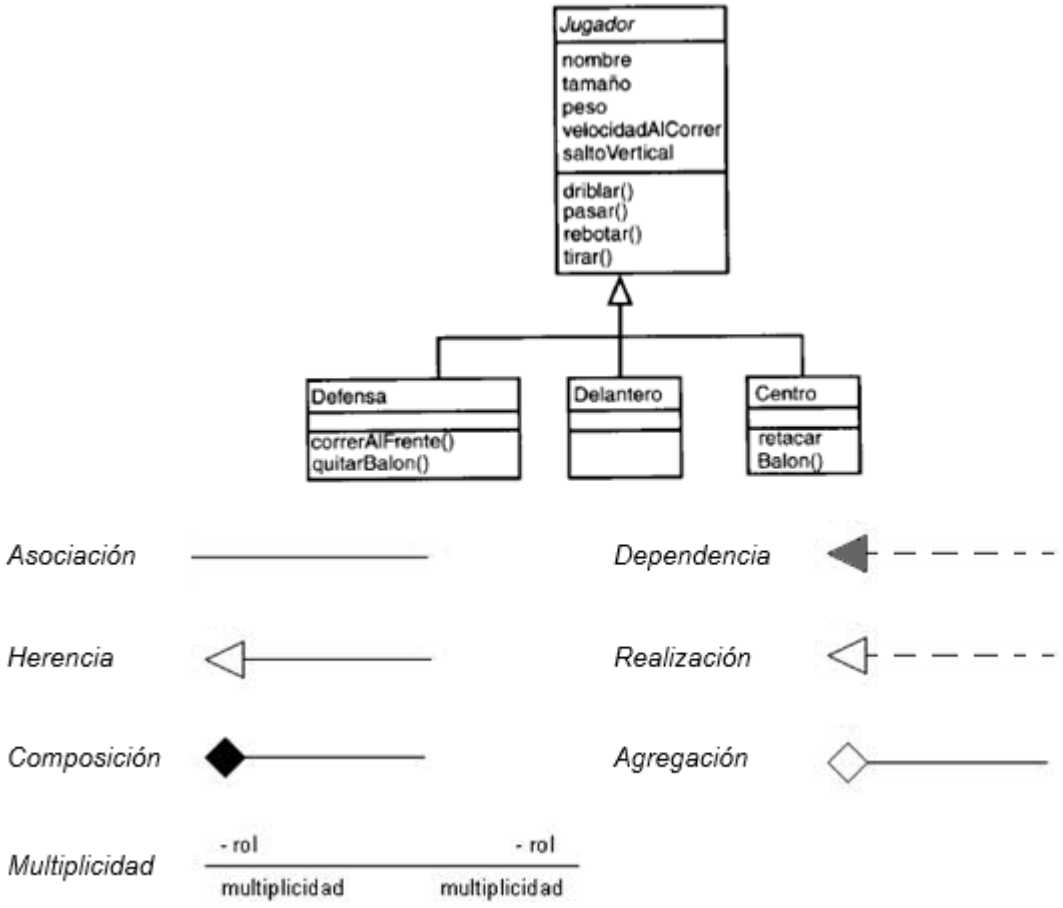


Ilustración 1. Diagrama de Clases

Fuente: Aprendiendo UML en 24 horas. Joseph Schmuller. Prentice Hall.

Los *Diagramas de Actividades* (*Ilustración 2 y 3*) muestran los pasos de una operación o un proceso. Las actividades son representadas con unos rectángulos con las esquinas redondeadas, y solo se pasa de una actividad a otra entre tanto el procesamiento de la actividad haya finalizado. En los diagramas de actividades también se pueden encontrar las famosas decisiones (Condicionales, Ilustración 2), las cuales a partir de un estado o respuesta toma una dirección u otra. Existen actividades que se pueden procesar de forma paralela, por ende existen las denominadas Rutas Concurrentes las cuales se representan con una línea horizontal gruesa para indicar el inicio y fin de las mismas (Ilustración 3). Otro elemento gráfico parten de indicaciones que buscan ejecutar otra actividad, dichas

indicaciones se representan con un pentágono convexo para el que envía, y un pentágono cóncavo para el que recibe.

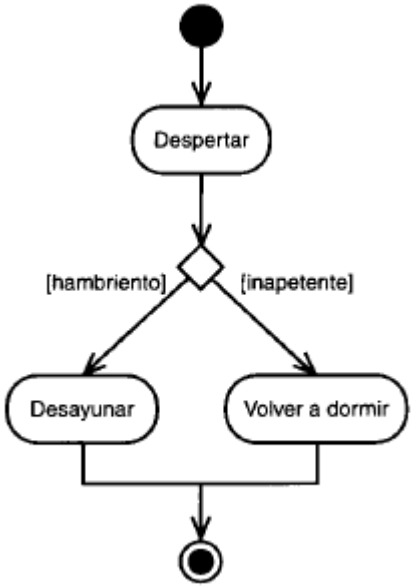


Ilustración 2. Diagrama de Actividades; Decisiones

Fuente: Aprendiendo UML en 24 horas. Joseph Schmuller. Prentice Hall.

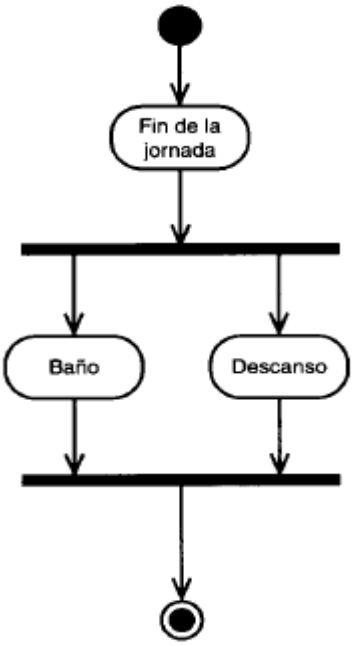


Ilustración 3. Diagrama de Actividades; Concurrencia

Fuente: Aprendiendo UML en 24 horas. Joseph Schmuller. Prentice Hall.

Los *Diagramas de Secuencias* muestran la forma en que los objetos se comunican entre sí al transcurrir el tiempo (Ilustración 4). En su representación se observan elementos como rectángulos alineados de forma horizontal en la parte superior del diagrama, los cuales representan a los objetos; Líneas discontinuas que se despenden perpendicularmente a cada objeto, la cual se conoce como línea de vida del objeto; sobre la línea de vida se sitúan rectángulos más pequeños conocidos como activación, los cuales representan la ejecución de una operación que realiza el objeto y la duración de la activación; finalmente se encuentran los mensajes representados en flechas que parten desde la línea de vida de un objeto hacia la línea de vida del mismo o hacia la de otro objeto (Ilustración 5), estas flechas pueden ser de tres formas diferentes en atención al tipo de mensaje que se pretenda mostrar: *Simple* para la transferencia de control de un objeto a otro,

Sincrónico que indica que esperará una respuesta a dicho mensaje antes de continuar con su trabajo, y *Asincrónico* el cual continua con su trabajo sin esperar respuesta al mensaje enviado.

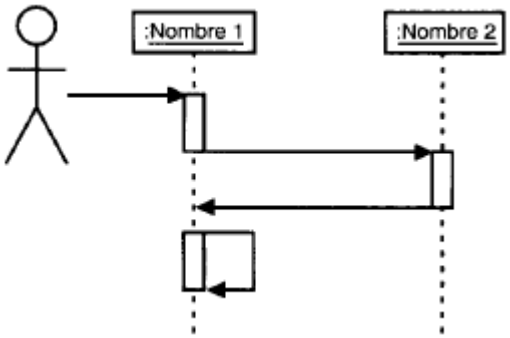


Ilustración 4. Diagrama de Secuencias

Fuente: Aprendiendo UML en 24 horas. Joseph Schmuller. Prentice Hall.

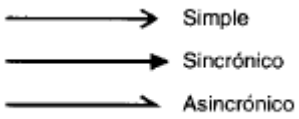


Ilustración 5. Tipos de Mensajes del Diagrama de Secuencias

Fuente: Aprendiendo UML en 24 horas. Joseph Schmuller. Prentice Hall.

Observando las ilustraciones anteriores de diagramas UML, podemos ver lo fácil que es visualizar, especificar, construir y documentar nuevos proyectos de software usando UML. Pero en nuestro caso ya existe una primera versión de TGENP, debido a lo cual se hace necesaria aplicar la Ingeniería Inversa.

Se puede definir la ingeniería inversa como "*El análisis de un sistema para identificar sus componentes actuales y las dependencias que existen entre ellos, para extraer y crear abstracciones de dicho sistema e información de su diseño*" [Chifofsky, 1990]. Otra definición habla sobre la ingeniera inversa como "*El proceso de analizar el código, documentación y comportamiento de un sistema para identificar sus componentes actuales y sus dependencias para extraer y crear una abstracción del sistema e información de diseño. El sistema en estudio no es alterado, si no que se produce conocimiento adicional acerca del mismo*" [SEI, 2004].

Según las definiciones anteriores, podemos ver a la Ingeniería Inversa como el proceso de extraer y crear abstracciones de un software usando su código fuente

como principal fuente de información, analizando su comportamiento e identificando sus componentes y las dependencias de los mismos.

Antes de comenzar el trabajo de ingeniería inversa como tal, es necesario que el código fuente “sucio” pase por un proceso de Reestructuración, donde éste se limpia y se restructura garantizando que todo el código esté orientado a la programación estructurada, haciendo más fácil la aplicación de la ingeniera inversa (Ilustración 6).

El núcleo o base de la ingeniería inversa es una actividad que se conoce como *extracción de abstracciones*, donde el ingeniero y/o analista debe analizar el software y en base a su código fuente extraer una especiación significativa de información, que varía dependiendo del tipo de ingeniería inversa que se esté aplicando; de datos, de lógica o procesamiento o de interfaces de usuario (Ilustración 6).

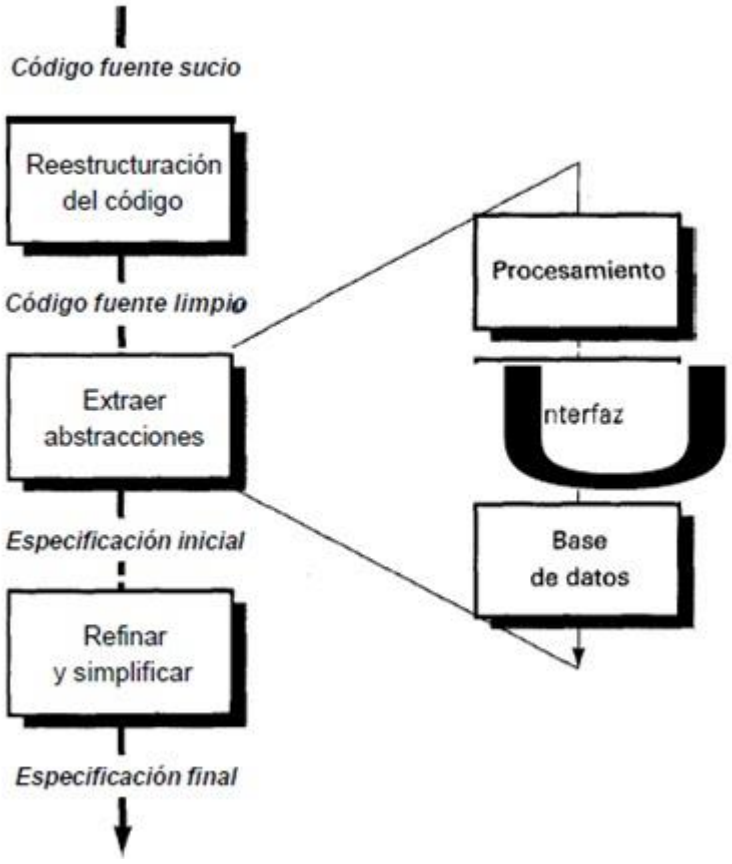


Ilustración 6. Proceso de Ingeniería Inversa

Fuente: Aprendiendo UML en 24 horas. Joseph Schmuller. Prentice Hall.

La ingeniería inversa de Datos se aplica sobre el código de base de datos (aplicación, SQL, etc.) para obtener los modelos relacionales o sobre los modelos relacionales para obtener el diagrama entidad-relación [3].

La Ingeniería Inversa de lógica o de proceso se aplica sobre el código de un programa para averiguar su lógica o sobre cualquier documento de diseño para obtener documentos de análisis o de requisitos [3].

Por último, la ingeniería inversa de interfaces de usuario se aplica con el objeto de mantener la lógica interna del programa para obtener los modelos y especificaciones que sirvieron de base para la construcción de la misma, con objeto de tomarlas como punto de partida en procesos de ingeniería directa que permitan modificar dicha interfaz [3].

3.2.1 Antecedentes

Acentuando la importancia que tiene la documentación frente al desarrollo y/o adaptación de un software, se traen a mención algunas experiencias basadas en la realización de modelos gráficos de UML, algunos aplicando el concepto de ingeniería inversa, permitiendo entender de forma mucho más fácil todos los aspectos trabajados en la codificación.

En el documento “Uso de la notación UML en el desarrollo de aplicaciones educativas” [4], los autores tratan de exponer su experiencia en cuanto al uso de UML en el proceso de desarrollo de aplicaciones educativas: “Este documento se presenta como la descripción de una experiencia en el uso de la notación del Lenguaje Unificado de Modelado o UML, con el propósito de diseñar aplicaciones educativas. La clásica cuestión de por qué realizamos el análisis y diseño si lo que el usuario necesita es que el software funcione, marca la línea entre lo que el desarrollador debe hacer y lo que el usuario final y quienes encargan el producto, piensan que debe tener programa” [4]. Los autores afirman que después de revisar sus fuentes y de experimentar la utilización de UML rescatan la gran capacidad de UML de vincular a los usuarios y desarrolladores del software. Además confirman que “La notación gráfica del UML posibilita dicha comunicación y su aprendizaje por parte de los integrantes del equipo multidisciplinario de desarrollo ajenos al área computacional toma poco tiempo”.

Estudiantes de la Universidad Autónoma de Baja California-México, plantearon en su trabajo de grado denominado *“Ingeniería Inversa y Reingeniería Aplicadas a Proyectos de Software Desarrollados por Alumnos de Nivel Licenciatura”,* la importancia de la documentación del software como herramienta que facilita la comprensión del código fuente y tratamientos que se pretendan realizar al mismo. La problemática planteada describe que a causa del entrenamiento recibido, los estudiantes de carreras de licenciaturas estaban capacitados para desarrollar software a nivel de escritura de código fuente, pero con poco nivel de análisis y diseño, sin incluir documentación útil en sus productos finales. En atención a esto se emplean procesos de ingeniería inversa y reingeniería a fin de obtener un porcentaje alto de la documentación de los aplicativos. Los resultados obtenidos fueron de provecho en la medida de que se generó un nivel adecuado de documentación, además de que en el análisis de los resultados obtenidos les permitió valorar las bondades del proceso de ingeniería, por ejemplo, en términos de orden, nivel de documentación y calidad del producto final.

En esta misma línea, otro grupo de estudiantes, pero esta vez de la Universidad Nacional Mayor de San Marcos – Perú, proponen en su tesis de grado una

metodología basada en ingeniería inversa a fin de recuperar la documentación funcional y las especificaciones de diseño de sistemas desarrollados con metodología orientada a objetos, de tal forma que se facilite el mantenimiento de los mismos. El proyecto “” argumenta la problemática de que *“Es casi seguro decir que el mantenimiento del software […] probablemente sea la parte más importante del ciclo de vida del software (aparte de ser la más olvidada). […] el mantenimiento […] incluyen el mejoramiento y la adición de nueva funcionalidad.”.* Además en este documento se expone el caso común en muchas empresas en los que se da rotación de personal (El desarrollador del código original no se encuentra y es tarea de nuevos desarrolladores realizar las tareas de mantenimiento). Como conclusiones este proyecto acentúa las bondades que ofrece la ingeniería inversa ante la falta de documentación de código fuente.

3.3 Objetivos

3.3.1 General

Modelar con UML el generador de código de aplicaciones web TGENP.

3.3.2 Específicos

- Aplicar Ingeniería Inversa al código fuente de TGENP a fin de generar diagramas automáticos.
- Crear Diagramas de Estructura.
    - Diagrama de Clases.
    - Diagrama de Paquetes.
    - Diagrama de Componentes.
- Crear Diagramas de Comportamiento.
    - Diagramas de casos de uso.
    - Diagramas de Actividades.
- Crear Diagramas de Iteración.
    - Diagramas de Secuencias.

3.4. Metodología

Al no encontrar una metodología estándar para la aplicación de ingeniería inversa a un software y en base al marco de trabajo que propone el Instituto de Ingeniería del Software (SEI 2004), se propone una metodología compuesta de las siguientes etapas:

3.4.1. Definición y delimitación del componente de software.

Es la primera etapa del proceso de ingeniería inversa y consiste en la selección y evaluación de los componentes a los que se les aplicará la ingeniería inversa.

3.4.2. Recolección de funcionalidades existentes.

- Diagrama y documentación de casos de uso: apoyados en el contexto de las especificaciones de requisitos y en los casos de uso que están implícitos en el software TGENP, los casos de uso permitirán evidenciar la trazabilidad que existe entre las especificaciones de requisitos y los fuentes.

3.4.3. Generación automática de los diagramas de clase y de paquetes mediante herramientas CASE.

- Diagrama de Clases: Se identificaran las clases que conforman la herramienta y las relaciones existentes entre las mismas, las cuales pueden ser de herencia, composición, agregación, asociación y uso mediante una herramienta CASE automatizada.

- Diagrama de Paquetes: Se identificarán las agrupaciones lógicas del software y las dependencias existentes entre estas agrupaciones.

3.4.4. Análisis de código fuente.

- Diagrama de actividades: Se describirá el proceso de generación de código como un flujo de trabajo a través de una serie de acciones, identificando en cada acción si la ejecuta una persona, un componente del software o un equipo.

- Diagrama de Secuencia: Se identificarán y describirán las iteraciones que se dan entre los objetos de la aplicación y los mensajes enviados y recibidos (por los objetos) durante el proceso de generación de código.

- Diagrama de componentes: Se identificaran los componentes que conforman del software y sus relaciones, lo cual permitirá mostrar la arquitectura física del software.

3.5 Resultados/Productos Esperados y Potenciales Beneficiarios

Tabla 1 Generación de nuevo conocimiento y/o nuevos desarrollos tecnológicos

Resultado/ Producto esperado	Indicador	Beneficiario
Diagrama de clases	1 diagrama	Comunidad de desarrolladores
Diagrama de paquetes	1 diagrama	Comunidad de desarrolladores
Diagrama de componentes	1 diagrama	Comunidad de desarrolladores
Diagrama de casos de uso	1 diagrama	Comunidad de desarrolladores
Documentación Casos de Uso	5 casos de uso	Comunidad de desarrolladores
Diagrama de actividades	2 diagrama	Comunidad de desarrolladores
Diagrama de secuencias	2 diagrama	Comunidad de desarrolladores

3.6 Impactos esperados a partir del uso de los resultados:

El principal impacto que busca causar este proyecto es el de evidenciar la importancia del uso de UML en las fases de análisis y diseño de los proyectos de Software. Al demostrar la importancia de UML se busca también tratar de implantar como buena práctica dentro del análisis y diseño de los proyectos de software la utilización de herramientas de modelado como UML. De esta forma cada integrante del equipo de desarrollo, y más aún cada nuevo integrante, podrán apreciar, entender y realizar los aportes correspondientes de una forma más acertada, aumentando la productividad del equipo y la calidad del producto resultante de cada proyecto.

3.7Cronograma de actividades

El cronograma de Actividades se creó en base a la metodología propuesta y se representó por medio de un diagrama de Gantt como se muestra en la siguiente imagen (Ilustración 7), usando la herramienta de software gratuita GanttProject:

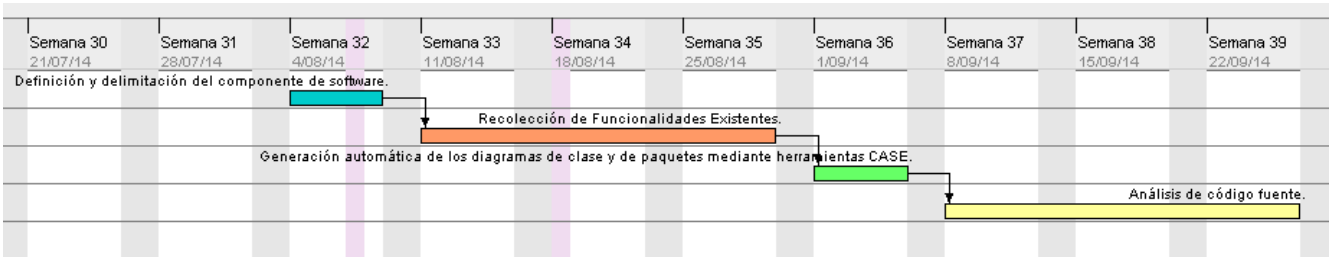


Ilustración 7. Diagrama de Gantt del Cronograma de Actividades

4. PRESUPUESTO

RUBROS GENERALES		RECURSOS PROPIOS		TOTAL
		EFECTIVO	(CAPACIDAD INSTALADA)	
Personal: - Servicio de 2 ingenieros por dos meses		4.000.000	0	4.000.000
Insumos: - Papelería - Impresiones - Fotocopias		0	150.000	150.000
Equipo	Compra - Computadores	0	3.400.000	3.400.000
	Arriendo	0	0	0
	Uso	0	0	0
Servicios técnicos:		0	0	0
Salidas de campo:		0	0	0
Viajes Nacionales, Internacionales y Cursos de entrenamiento:		0	0	0
Software:		0	0	0
Realización talleres, foros:		0	0	0
Material bibliográfico especializado:		0	0	0
Publicaciones y patentes: - 2 Argollado - 2 Impresiones		140.000	0	140.000

- 2 Grabaciones en CD Rotulados			
Imprevistos	300.000	0	300.000
TOTAL	**4.440.000**	**3.550.000**	**7.990.000**

5. RESULTADOS

Definición y delimitación del componente de software

En esta etapa se definieron los componentes a tener en cuenta en los diagramas UML, las principales funcionalidades de sistema, los componentes lógicos y físicos que utilizan estas funcionalidades y la arquitectura MVC con la cual está desarrollado.

En primera instancia se miraron las funcionalidades que ofrece este software a fin de delimitar los componentes de tal forma que se ofrezca una documentación acertada y fácilmente entendible por demás desarrolladores.

La interfaz de TGENP ofrece al usuario en primera instancia la posibilidad de seleccionar la base de datos y las directrices de acceso a la misma, seguidamente luego de realizar una validación y autenticación previa ofrece al usuario la posibilidad de escoger entre múltiples opciones, las cuales concluyen en generar todo el código para el nuevo proyecto, o simplemente generar código parcial desde tipos de contenido o exclusiones de tablas para la base de datos. TGENP cubre aspectos tales del desarrollo que brinda al desarrollador la facilidad de luego de haber realizado la creación de su base de datos, obtener todos los métodos básicos, vistas, modelos y demás código en sólo 3 pasos. Atendiendo esto, nos queda detallar ese proceso macro de desarrollo completo del código, en cuanto a su generación desde una base de datos debidamente elaborada, y ahondar un poco en cuanto a la conexión e interfaz que se tiene con el usuario.

Recolección de funcionalidades existentes

El diagrama de casos de uso permite evidenciar la trazabilidad que existe entre las especificaciones de requisitos implícitos en el generador y las fuentes. Para la creación del diagrama de casos de uso de la ilustración 8, se usó la herramienta de software gratuita StartUML y para su documentación se utilizó la herramienta de software experimental de Gestión de requisitos REM (Requirements Management).

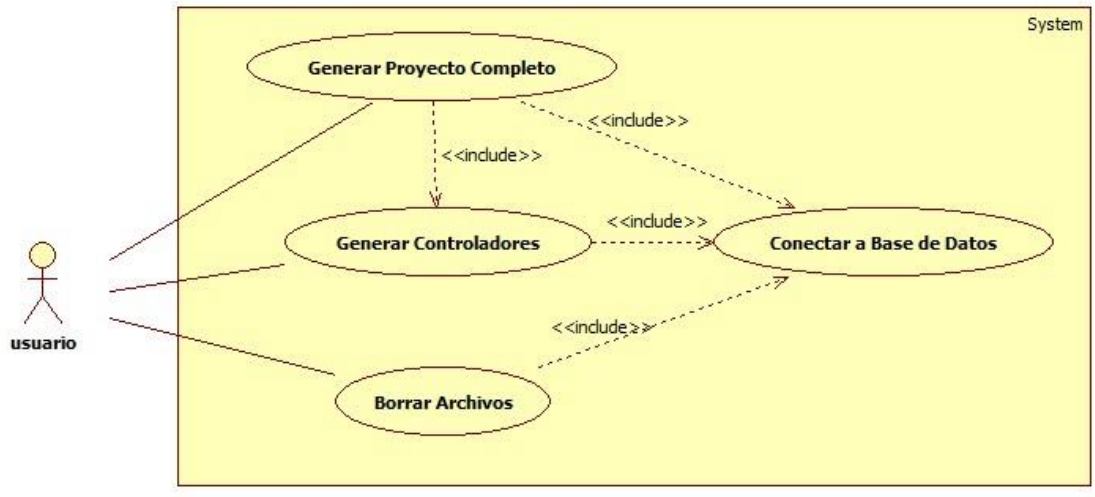


Ilustración 8. Diagrama de Casos de Uso

1 Organizaciones

Organización	**Estudiantes de Grado**
Dirección	Calle 24 N. 18a-57 Santa Marta D.T.C.H. - Colombia
Teléfono	3045244545

Organización	**Universidad del Magdalena**
Dirección	Carrera 32 No 22 - 08 Santa Marta D.T.C.H. - Colombia
Teléfono	(57 - 5) 4217940-4301292

2 Participantes

Participante	**Carlos Adarraga**
Organización	Freelance
Rol	Desarrollador
Es desarrollador	No
Es cliente	No
Es usuario	Sí
Comentarios	Ingeniero de Sistemas

Participante	**Carlos Oliveros**
Organización	Freelance
Rol	Desarrollador
Es desarrollador	Sí
Es cliente	No

Es usuario	Sí
Comentarios	Ingeniero de Sistemas

3 Requisitos Funcionales

FRQ-0001	**Conexión a Base de Datos**
Versión	1.0 (03/08/2014)
Autores	Carlos Adarraga Carlos Oliveros
Fuentes	Carlos Oliveros
Dependencias	• [UC-0002] Conectar a Base de Datos
Descripción	El sistema deberá *conectarse a la base de datos MYSQL de la cual se va a generar código fuente PHP, usando como datos de acceso la dirección IP o hostname del servidor MYSQL, el nombre de la base de datos, un usuario y clave válidos.*
Importancia	vital
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

FRQ-0002	**Generar Proyecto Completo**
Versión	1.0 (03/08/2014)
Autores	Carlos Adarraga Carlos Oliveros
Fuentes	Carlos Oliveros
Dependencias	Ninguno
Descripción	El sistema deberá *generar todo el código fuente PHP del proyecto, con la arquitectura de desarrollo MVC.*
Importancia	vital
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

FRQ-0003	**Generar Modelos**
Versión	1.0 (03/08/2014)
Autores	Carlos Adarraga

	Carlos Oliveros
Fuentes	Carlos Oliveros
Dependencias	Ninguno
Descripción	El sistema deberá *generar los modelos del proyecto.*
Importancia	importante
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

FRQ-0004	**Generar Vistas**
Versión	1.0 (03/08/2014)
Autores	Carlos Adarraga Carlos Oliveros
Fuentes	Carlos Oliveros
Dependencias	Ninguno
Descripción	El sistema deberá *generar las vistas del proyecto.*
Importancia	importante
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

FRQ-0005	**Generar Controladores**
Versión	1.0 (03/08/2014)
Autores	Carlos Adarraga Carlos Oliveros
Fuentes	Carlos Oliveros
Dependencias	Ninguno
Descripción	El sistema deberá *generar los controladores del proyecto.*
Importancia	importante
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

FRQ-0006	Generar WEB
Versión	1.0 (03/08/2014)
Autores	Carlos Adarraga Carlos Oliveros
Fuentes	Carlos Oliveros
Dependencias	Ninguno
Descripción	El sistema deberá *generar y/o copiar todos los archivos webs del proyecto como estilos CSS, archivos con funcionalidades JavaScripts y el framework Jquery*
Importancia	importante
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

FRQ-0007	Generar Config
Versión	1.0 (03/08/2014)
Autores	Carlos Adarraga Carlos Oliveros
Fuentes	Carlos Oliveros
Dependencias	Ninguno
Descripción	El sistema deberá *generar todo el código fuente de las clases principales de configuración del proyecto: controlador, Modelo y Vista padres, clase de herramientas y de App, clases de conexión a Base de Datos, manejadores de entorno de trabajo.*
Importancia	vital
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

FRQ-0008	Generar Librerías
Versión	1.0 (03/08/2014)
Autores	Carlos Adarraga Carlos Oliveros
Fuentes	Carlos Oliveros
Dependencias	Ninguno

Descripción	El sistema deberá *generar y/o copiar las librerías de apoyo como PHPExcel, R&os PDF, PclZip, PHPMailer*
Importancia	importante
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

FRQ-0009	**Generar Datos Iniciales**
Versión	1.0 (03/08/2014)
Autores	Carlos Adarraga Carlos Oliveros
Fuentes	Carlos Oliveros
Dependencias	Ninguno
Descripción	El sistema deberá *crear una base de datos central, donde se registrara el proyecto (sistema) generado, un usuario por defecto, y un rol de superadmin con una de estructura XML de navegación-permisos para el usuario y el proyecto registrados.*
Importancia	vital
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

FRQ-0010	**Generar Index**
Versión	1.0 (03/08/2014)
Autores	Carlos Adarraga Carlos Oliveros
Fuentes	Carlos Oliveros
Dependencias	Ninguno
Descripción	El sistema deberá *generar el index principal del proyecto.*
Importancia	quedaría bien
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

FRQ-0011	**Borrar Código**
Versión	1.0 (03/08/2014)

Autores	Carlos Adarraga Carlos Oliveros
Fuentes	Carlos Oliveros
Dependencias	Ninguno
Descripción	El sistema deberá *tener una opción para borrar los archivos generados y/o copiados.*
Importancia	quedaría bien
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

FRQ-0012	**Salir**
Versión	1.0 (03/08/2014)
Autores	Carlos Adarraga Carlos Oliveros
Fuentes	Carlos Oliveros
Dependencias	Ninguno
Descripción	El sistema deberá *tener una opción para salir del sistema*
Importancia	quedaría bien
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

4 Actores

ACT-0001	**Usuario**
Versión	1.0 (03/08/2014)
Autores	Carlos Adarraga Carlos Oliveros
Fuentes	Carlos Adarraga Carlos Oliveros
Descripción	Este actor representa *al usuario del sistema*
Comentarios	Ninguno

5 Casos de Uso

UC-0002	Conectar a Base de Datos	
Versión	1.0 (03/08/2014)	
Autores	Carlos Adarraga Carlos Oliveros	
Fuentes	Carlos Oliveros	
Dependencias	• [FRQ-0001] Conexión a Base de Datos	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *Se acceda al sistema*	
Precondición	Que exista la base de datos normalizada creada en el servidor MYSQL.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-0001) *Ingresa los parámetros de conexión a base de datos: IP o Hostname de servidor MYSQL, nombre de base de datos, usuario root y clave de MYSQL y hace clic en el botón "Entrar" (Ilustración 9).*
	2	El sistema *valida los parámetros de conexión y muestra la pantalla de bienvenida con el menú funciones (Ilustración 10).*
Postcondición	Que la base de datos tenga tablas creadas	
Excepciones	**Paso**	**Acción**
	2	Si *no se ingresa alguno de los parámetros de conexión,* el sistema *muestra los mensajes de validación en cada uno de los parámetros (campos) (Ilustración 11)*, a continuación este caso de uso *queda sin efecto*
	2	Si *el servidor, o el usuario root o la clave no son válidos,* el sistema *muestra el mensaje de error "Error de conexión al servidor de base de datos" (Ilustración 12)*, a continuación este caso de uso *queda sin efecto*
	2	Si *la base de datos no existe,* el sistema *muestra el mensaje de error "No existe la base de datos seleccionada" (Ilustración 13)*, a continuación este caso de uso *queda sin efecto*
Importancia	vital	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0003	Generar proyecto completo
Versión	1.0 (03/08/2014)
Autores	Carlos Adarraga Carlos Oliveros
Fuentes	Carlos Oliveros

Dependencias	• [FRQ-0002] Generar Proyecto Completo	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *Clic en opción "Generar todo"*	
Precondición	Caso de uso "Conectar a Base de Datos"	
Secuencia normal	**Paso**	**Acción**
	1	El sistema *muestra las tablas de la base de datos, una opción de "generar estructura de rol" y las tablas de la base de datos central (SAS) si existe (Ilustración 14).*
	2	El actor Usuario (ACT-0001) *selecciona las tablas a las cuales se va a generar el código y la opción "generar estructura de rol" si lo requiere y hace clic en "Generar"*
	3	El sistema *muestra una imagen animada (gif) de "carga" o "loading" mientras genera el proyecto (Ilustración 15)*
	4	El sistema *filtra las tablas seleccionadas*
	5	El sistema *obtiene los campos de cada tabla*
	6	El sistema *genera el modelo de cada tabla*
	7	El sistema *genera el controlador de cada tabla*
	8	El sistema *genera las vistas de cada tabla*
	9	El sistema *genera el index principal*
	10	El sistema *genera el config*
	11	El sistema *copia los archivos web*
	12	Si *no existen,* el sistema *crea las tablas y registros iniciales en la base de datos central (tablas: sistemas, roles, usuarios, roles_usuarios)*
	13	El sistema *copia las librerías*
	14	El sistema *muestra los archivos creados (vistas, controladores, modelos, index, archivos de configuración), los archivos copiados (css, js, flash, media) y las tablas y registros creados de la base de datos central (sistemas, usuarios, roles, manuales, roles_sistemas) de color verde (Ilustración 16).*
Postcondición	N/A	
Excepciones	**Paso**	**Acción**
	5	Si *una tabla no tiene campos,* el sistema *muestra un mensaje indicando que "la tabla no posee campos"*, a continuación este caso de uso *continúa*
	14	Si *alguna tabla no tiene campos,* el sistema *muestra un mensaje de color naranja indicando que "La tabla 'X' no posee campos" (Ilustración 17)*, a continuación este caso de uso *continúa*
Importancia	vital	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0004	**Generar Controladores**

Versión	1.0 (08/09/2014)
Autores	Carlos Adarraga Carlos Oliveros
Fuentes	Carlos Oliveros
Dependencias	• [FRQ-0005] Generar Controladores
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *Clic en opción "Generar controladores"*
Precondición	Caso de uso "Conectar a Base de Datos"

Secuencia normal	**Paso**	**Acción**
	1	El sistema *muestra las tablas de la base de datos y las tablas de la base de datos central (SAS) si existe (Ilustración 18).*
	2	El actor Usuario (ACT-0001) *selecciona las tablas a las cuales se les va a generar los controladores y hace clic en "Generar"*
	3	El sistema *muestra una imagen animada (gif) de "carga" o "loading" mientras genera los controladores (Ilustración 15)*
	4	El sistema *muestra mensajes de color verde con los controladores creados (Ilustración 19)*

Postcondición	N/A

Excepciones	**Paso**	**Acción**
	4	Si *alguna tabla no tiene campos,* el sistema *muestra un mensaje de color naranja indicando que "La tabla X no posee campos" (Ilustración 20)*, a continuación este caso de uso *continúa*

Importancia	vital
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

UC-0005	**Borrar todo**
Versión	1.0 (08/09/2014)
Autores	Carlos Adarraga Carlos Oliveros
Fuentes	Carlos Adarraga Carlos Oliveros
Dependencias	• [FRQ-0011] Borrar Código
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se haga Clic en la opción "Borrar Todo"*
Precondición	Caso de uso "Conectar a Base de Datos"

Secuencia normal	**Paso**	**Acción**
	1	El sistema *muestra el mensaje "Directorio "result_generated" borrado con éxito." (Ilustración 21)*
Postcondición	N/A	
Excepciones	**Paso**	**Acción**
	1	Si *ocurre un error,* el sistema *muestra el mensaje "El directorio "result_generate" no fue borrado" (Ilustración 22)*, a continuación este caso de uso *queda sin efecto*
Importancia	vital	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0006	**Salir**
Versión	1.0 (08/09/2014)
Autores	Carlos Adarraga Carlos Oliveros
Fuentes	Carlos Adarraga Carlos Oliveros
Dependencias	• [FRQ-0012] Salir
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se haga Clic en la opción "Salir"*
Precondición	Caso de uso "Conectar a Base de Datos"

Secuencia normal	**Paso**	**Acción**
	1	El sistema *cierra sesión y muestra la vista de Conexión a Base de Datos (Ilustración 9)*
Postcondición	N/A	
Excepciones	**Paso**	**Acción**
	1	Si *ocurre un error durante el proceso de Salir,* el sistema *muestra el mensaje de error "No se pudo cerrar sesión" (Ilustración 23)*, a continuación este caso de uso *queda sin efecto*
Importancia	vital	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

6 Figuras

Ilustración 9. Conexión a Base de Datos

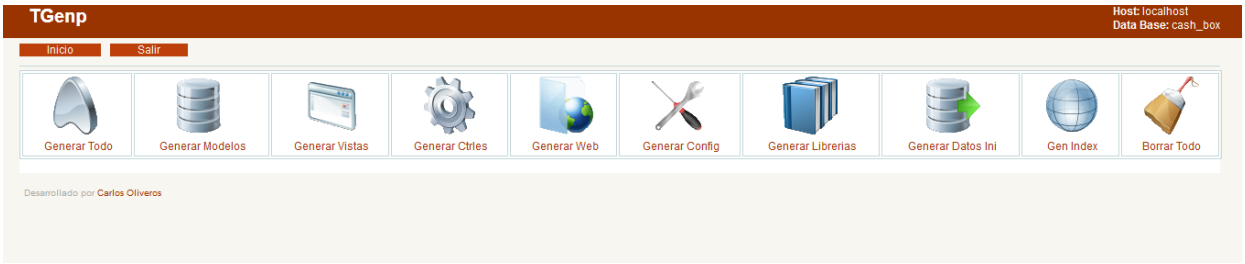


Ilustración 10. Bienvenida TGENP

Ilustración 11. Validar datos conexión

Ilustración 12. Error 1 conexión a BD

Ilustración 13. Error 2 conexión a BD

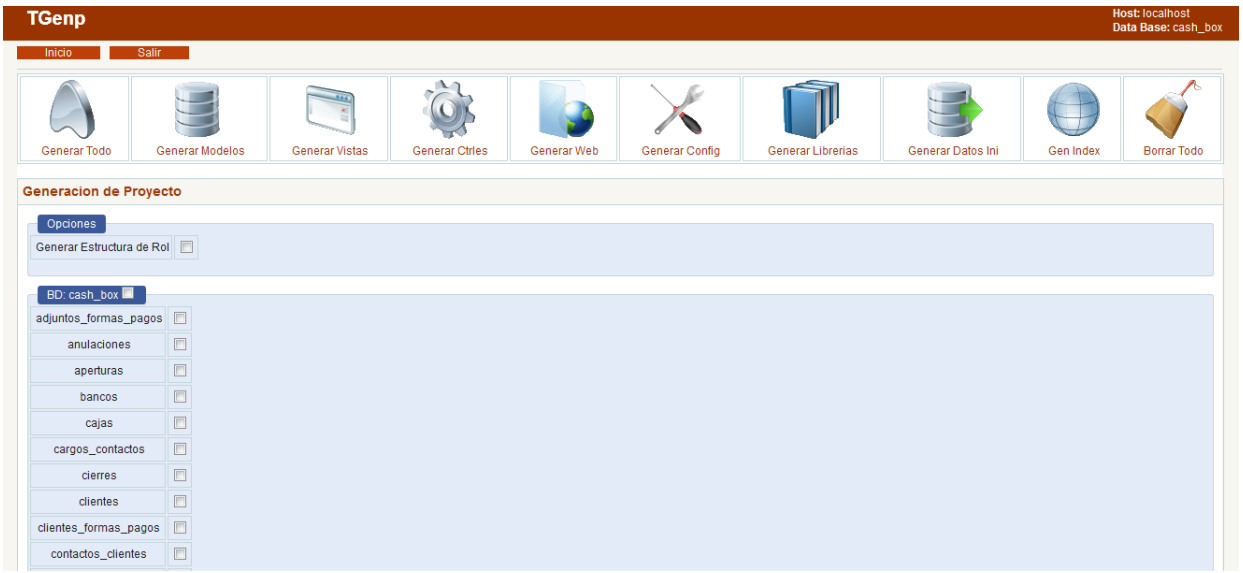


Ilustración 14. Selección de Tablas a crear

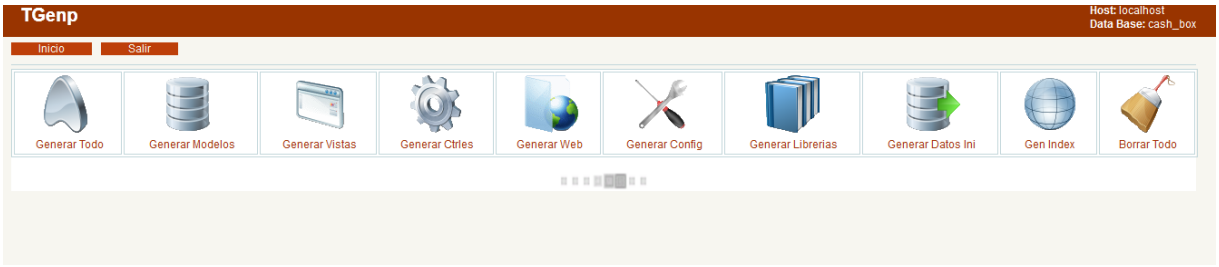


Ilustración 15. Generación de Proyecto

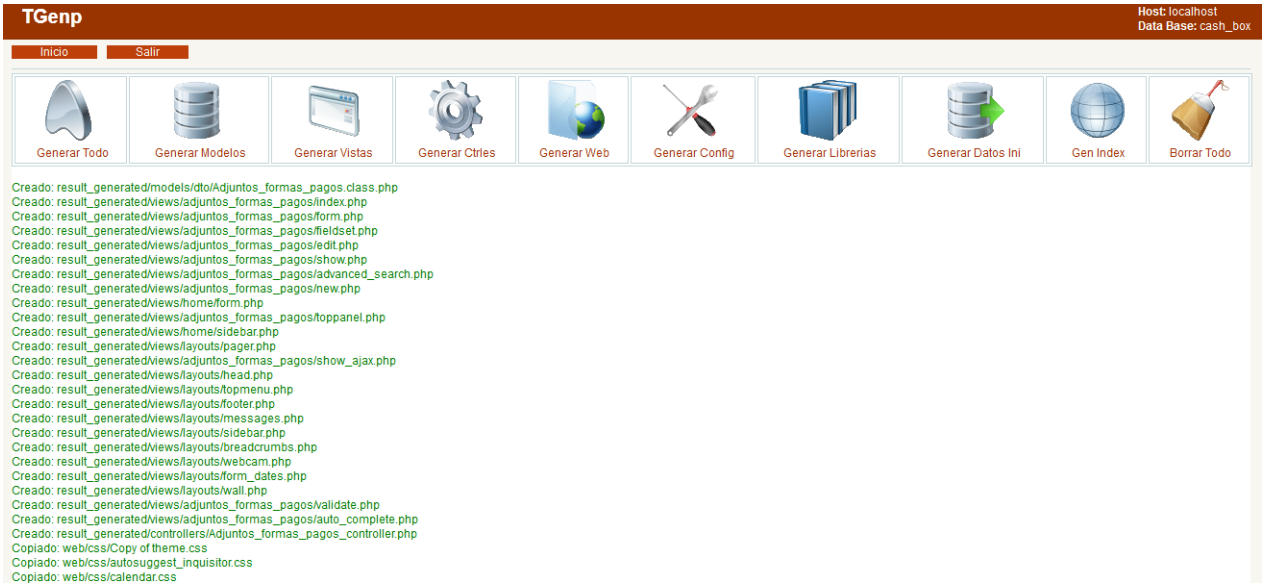


Ilustración 16. Resultado de Generación de Proyecto Completo

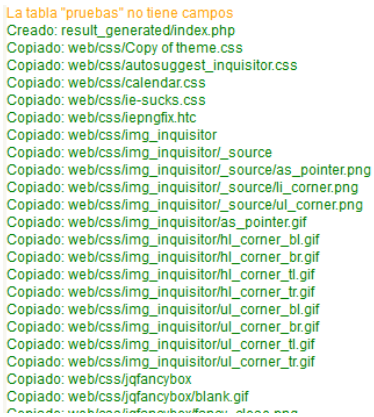


Ilustración 17. Mensaje tabla sin campos

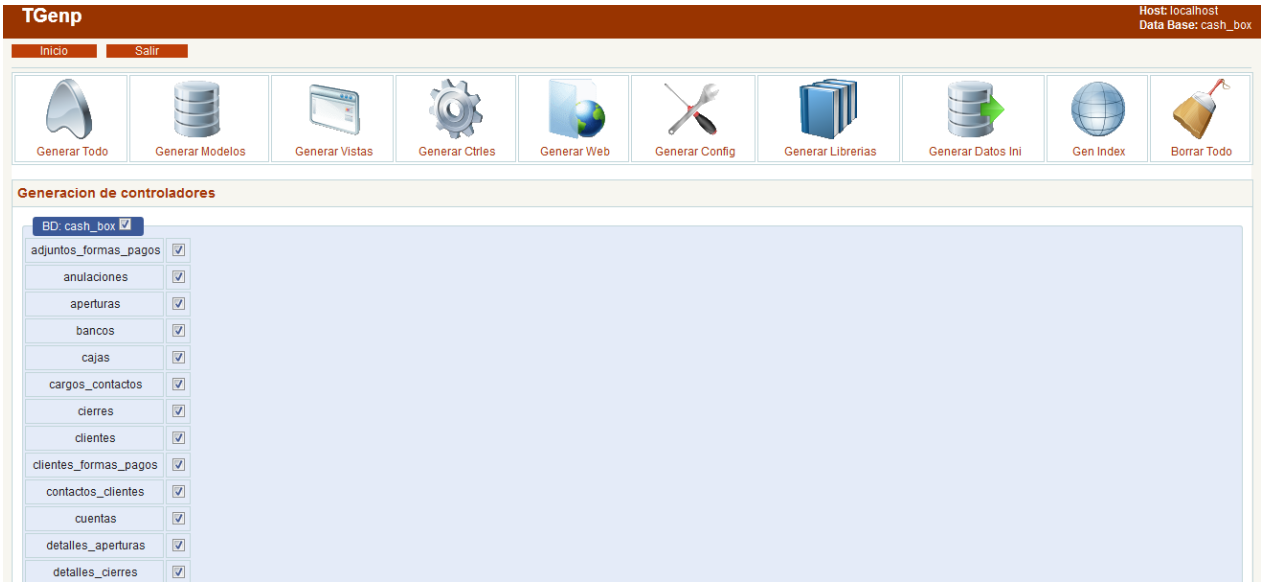


Ilustración 18. Lista de tablas para generar controladores

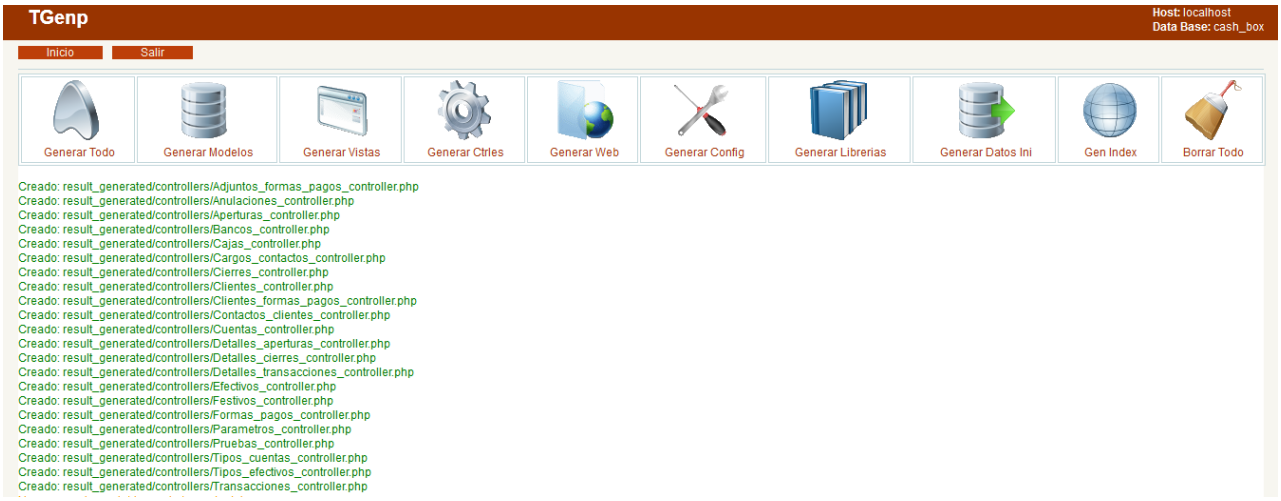


Ilustración 19. Mensajes con controladores creados

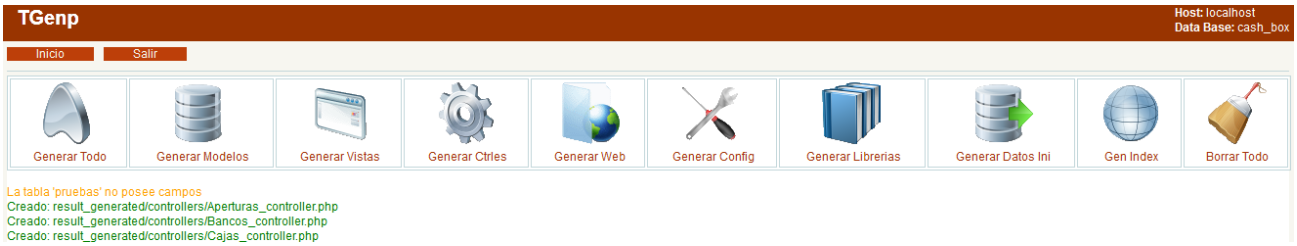


Ilustración 20. Mensaje "Tabla no posee campos" para generación de controladores

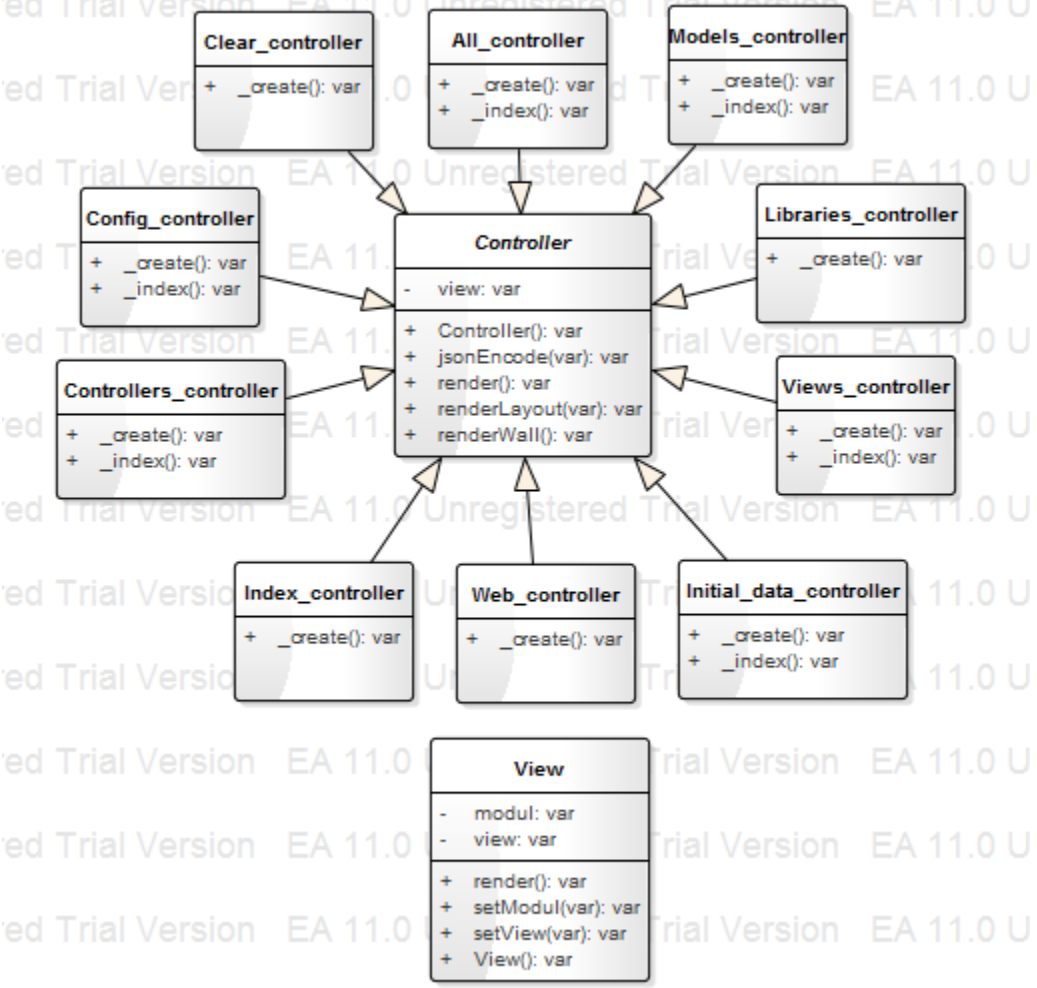
Ilustración 21. Borrar todo

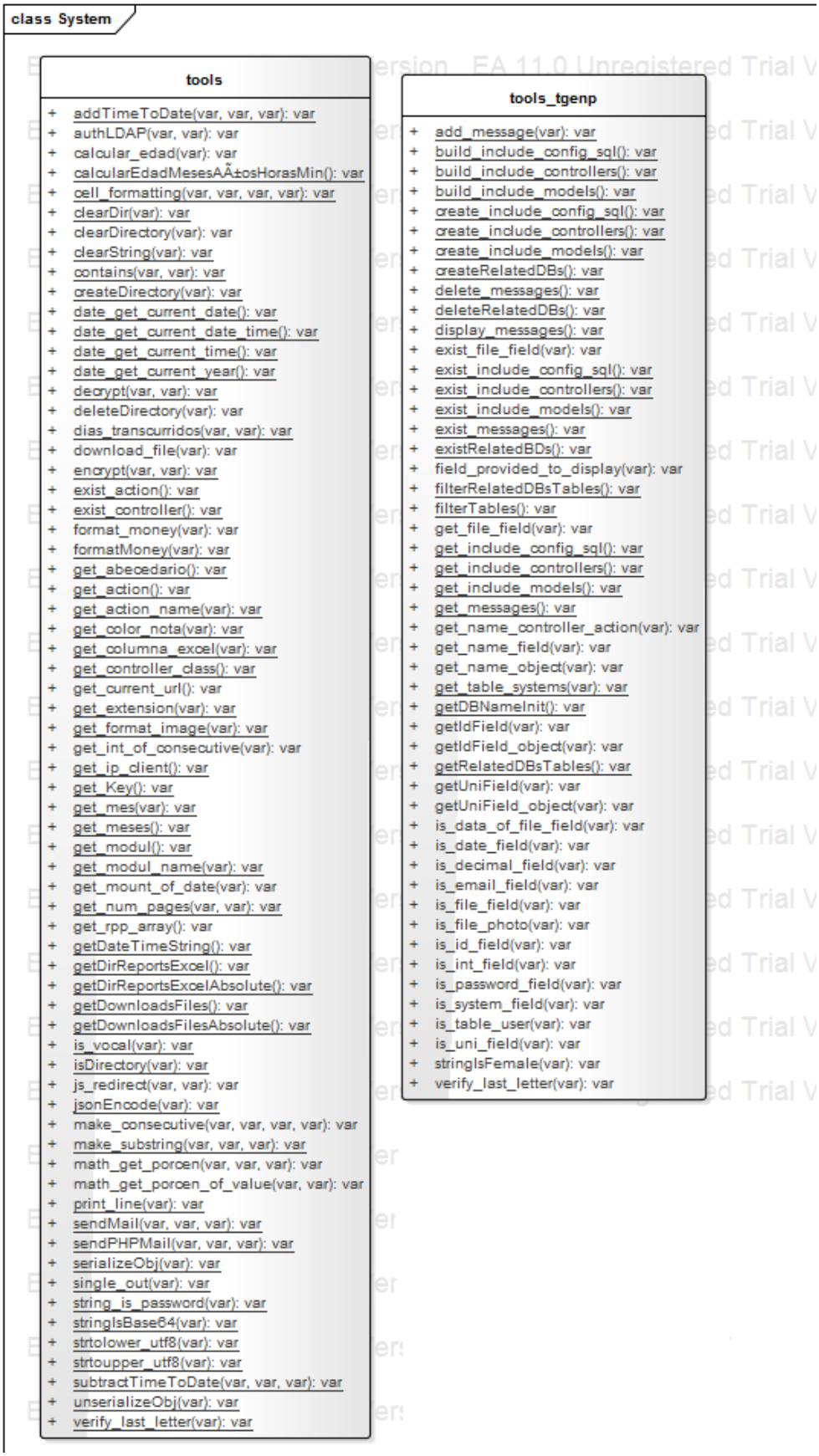
Ilustración 22. Error Borrar Todo

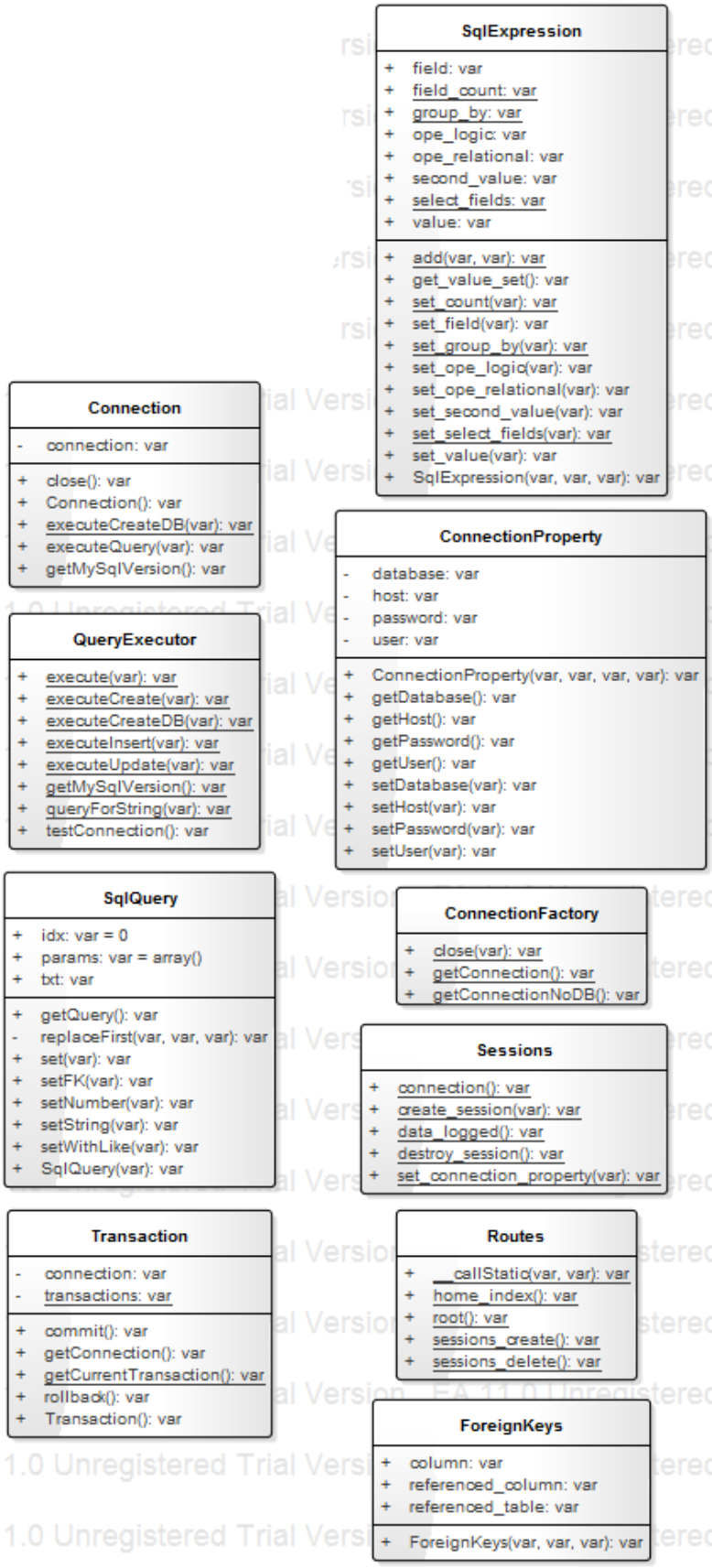
Ilustración 23. Error Salir

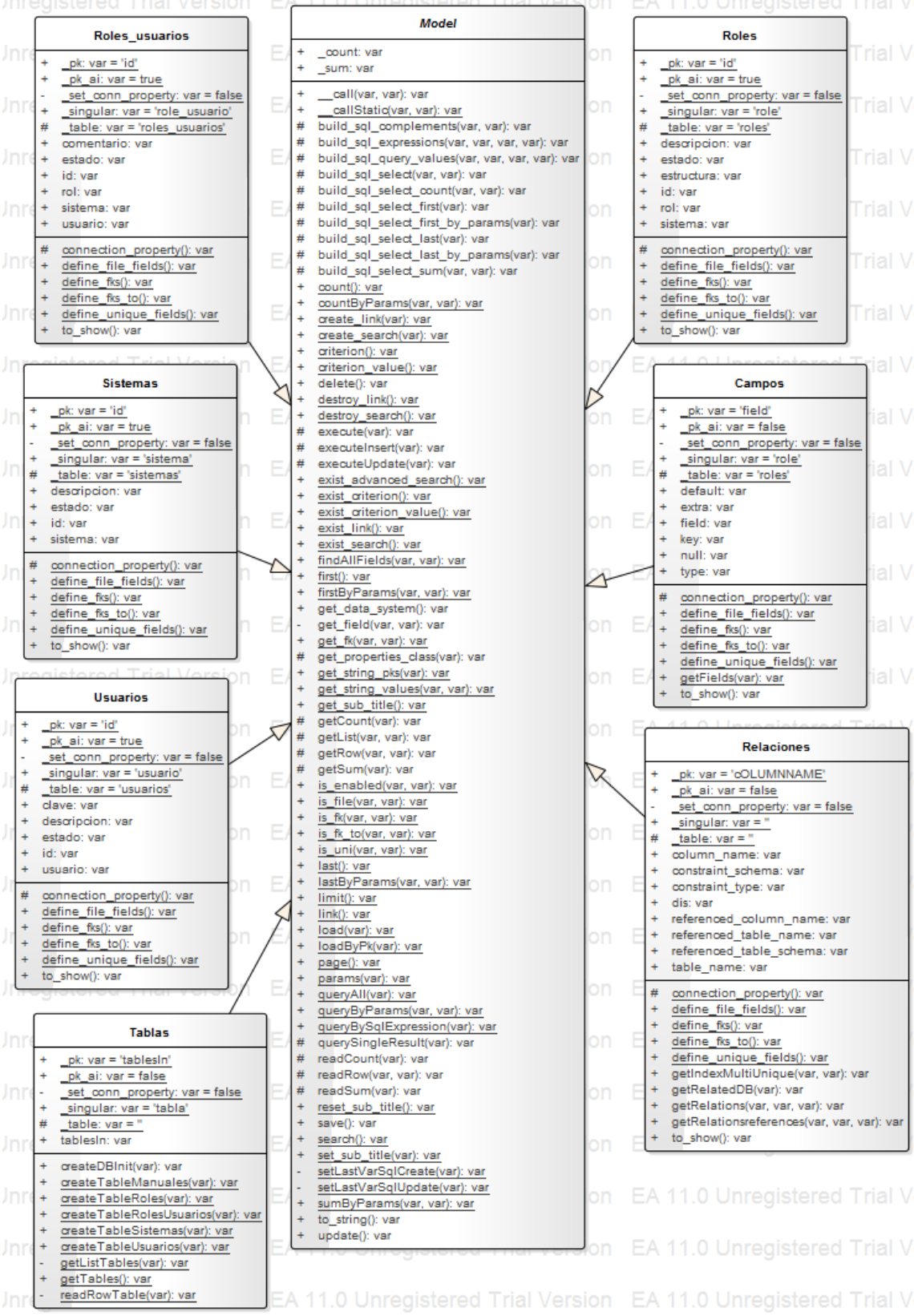
Generación automática de los diagramas de clase y de paquetes mediante herramientas CASE

Para la creación del Diagrama de Clases se usó el software Enterprise Architect en su periodo de prueba, pues no es una herramienta gratuita. El software permitió generar el diagrama de clases a partir del código fuente de TGENP de forma automática, aplicando ingeniería inversa, identificando 41 clases entre las cuales se pueden mencionar la clase Model y la clase Controller, de las cuales heredan todos los modelos y controladores respectivamente, y las nueve clases que conforman el paquete generator.


Clear_controller
+ _create(): var
All_controller
+ _create(): var
+ _index(): var
Models_controller
+ _create(): var
+ _index(): var
Config_controller
+ _create(): var
+ _index(): var
Controller
- view: var
+ Controller(): var
+ jsonEncode(var): var
+ render(): var
+ renderLayout(var): var
+ renderWall(): var
Libraries_controller
+ _create(): var
Controllers_controller
+ _create(): var
+ _index(): var
Views_controller
+ _create(): var
+ _index(): var
Index_controller
+ _create(): var
Web_controller
+ _create(): var
Initial_data_controller
+ _create(): var
+ _index(): var
View
- modul: var
- view: var
+ render(): var
+ setModul(var): var
+ setView(var): var
+ View(): var


class System
tools
+ addTimeToDate(var, var, var): var
+ authLDAP(var, var): var
+ calcular_edad(var): var
+ calcularEdadMesesAÃ±osHorasMin(): var
+ cell_formatting(var, var, var, var): var
+ clearDir(var): var
+ clearDirectory(var): var
+ clearString(var): var
+ contains(var, var): var
+ createDirectory(var): var
+ date_get_current_date(): var
+ date_get_current_date_time(): var
+ date_get_current_time(): var
+ date_get_current_year(): var
+ decrypt(var, var): var
+ deleteDirectory(var): var
+ dias_transcurridos(var, var): var
+ download_file(var): var
+ encrypt(var, var): var
+ exist_action(): var
+ exist_controller(): var
+ format_money(var): var
+ formatMoney(var): var
+ get_abecedario(): var
+ get_action(): var
+ get_action_name(var): var
+ get_color_nota(var): var
+ get_columna_excel(var): var
+ get_controller_class(): var
+ get_current_url(): var
+ get_extension(var): var
+ get_format_image(var): var
+ get_int_of_consecutive(var): var
+ get_ip_client(): var
+ get_Key(): var
+ get_mes(var): var
+ get_meses(): var
+ get_modul(): var
+ get_modul_name(var): var
+ get_mount_of_date(var): var
+ get_num_pages(var, var): var
+ get_rpp_array(): var
+ getDateTimeString(): var
+ getDirReportsExcel(): var
+ getDirReportsExcelAbsolute(): var
+ getDownloadsFiles(): var
+ getDownloadsFilesAbsolute(): var
+ is_vocal(var): var
+ isDirectory(var): var
+ js_redirect(var, var): var
+ jsonEncode(var): var
+ make_consecutive(var, var, var, var): var
+ make_substring(var, var, var): var
+ math_get_porcen(var, var, var): var
+ math_get_porcen_of_value(var, var): var
+ print_line(var): var
+ sendMail(var, var, var): var
+ sendPHPMail(var, var, var): var
+ serializeObj(var): var
+ single_out(var): var
+ string_is_password(var): var
+ stringIsBase64(var): var
+ strtolower_utf8(var): var
+ strtoupper_utf8(var): var
+ subtractTimeToDate(var, var, var): var
+ unserializeObj(var): var
+ verify_last_letter(var): var
tools_tgenp
+ add_message(var): var
+ build_include_config_sql(): var
+ build_include_controllers(): var
+ build_include_models(): var
+ create_include_config_sql(): var
+ create_include_controllers(): var
+ create_include_models(): var
+ createRelatedDBs(): var
+ delete_messages(): var
+ deleteRelatedDBs(): var
+ display_messages(): var
+ exist_file_field(var): var
+ exist_include_config_sql(): var
+ exist_include_controllers(): var
+ exist_include_models(): var
+ exist_messages(): var
+ existRelatedBDs(): var
+ field_provided_to_display(var): var
+ filterRelatedDBsTables(): var
+ filterTables(): var
+ get_file_field(var): var
+ get_include_config_sql(): var
+ get_include_controllers(): var
+ get_include_models(): var
+ get_messages(): var
+ get_name_controller_action(var): var
+ get_name_field(var): var
+ get_name_object(var): var
+ get_table_systems(var): var
+ getDBNameInit(): var
+ getIdField(var): var
+ getIdField_object(var): var
+ getRelatedDBsTables(): var
+ getUniField(var): var
+ getUniField_object(var): var
+ is_data_of_file_field(var): var
+ is_date_field(var): var
+ is_decimal_field(var): var
+ is_email_field(var): var
+ is_file_field(var): var
+ is_file_photo(var): var
+ is_id_field(var): var
+ is_int_field(var): var
+ is_password_field(var): var
+ is_system_field(var): var
+ is_table_user(var): var
+ is_uni_field(var): var
+ stringIsFemale(var): var
+ verify_last_letter(var): var


SqlExpression
+ field: var
+ field_count: var
+ group_by: var
+ ope_logic: var
+ ope_relational: var
+ second_value: var
+ select_fields: var
+ value: var
+ add(var, var): var
+ get_value_set(): var
+ set_count(var): var
+ set_field(var): var
+ set_group_by(var): var
+ set_ope_logic(var): var
+ set_ope_relational(var): var
+ set_second_value(var): var
+ set_select_fields(var): var
+ set_value(var): var
+ SqlExpression(var, var, var): var
Connection
- connection: var
+ close(): var
+ Connection(): var
+ executeCreateDB(var): var
+ executeQuery(var): var
+ getMySqlVersion(): var
ConnectionProperty
- database: var
- host: var
- password: var
- user: var
+ ConnectionProperty(var, var, var, var): var
+ getDatabase(): var
+ getHost(): var
+ getPassword(): var
+ getUser(): var
+ setDatabase(var): var
+ setHost(var): var
+ setPassword(var): var
+ setUser(var): var
QueryExecutor
+ execute(var): var
+ executeCreate(var): var
+ executeCreateDB(var): var
+ executeInsert(var): var
+ executeUpdate(var): var
+ getMySqlVersion(var): var
+ queryForString(var): var
+ testConnection(): var
SqlQuery
+ idx: var = 0
+ params: var = array()
+ txt: var
+ getQuery(): var
- replaceFirst(var, var, var): var
+ set(var): var
+ setFK(var): var
+ setNumber(var): var
+ setString(var): var
+ setWithLike(var): var
+ SqlQuery(var): var
ConnectionFactory
+ close(var): var
+ getConnection(): var
+ getConnectionNoDB(): var
Sessions
+ connection(): var
+ create_session(var): var
+ data_logged(): var
+ destroy_session(): var
+ set_connection_property(var): var
Transaction
- connection: var
- transactions: var
+ commit(): var
+ getConnection(): var
+ getCurrentTransaction(): var
+ rollback(): var
+ Transaction(): var
Routes
+ __callStatic(var, var): var
+ home_index(): var
+ root(): var
+ sessions_create(): var
+ sessions_delete(): var
ForeignKeys
+ column: var
+ referenced_column: var
+ referenced_table: var
+ ForeignKeys(var, var, var): var


Roles_usuarios
+ _pk: var = 'id'
+ _pk_ai: var = true
- _set_conn_property: var = false
+ _singular: var = 'role_usuario'
# _table: var = 'roles_usuarios'
+ comentario: var
+ estado: var
+ id: var
+ rol: var
+ sistema: var
+ usuario: var
# connection_property(): var
+ define_file_fields(): var
+ define_fks(): var
+ define_fks_to(): var
+ define_unique_fields(): var
+ to_show(): var
Model
+ _count: var
+ _sum: var
+ __call(var, var): var
+ __callStatic(var, var): var
# build_sql_complements(var, var): var
# build_sql_expressions(var, var, var, var): var
# build_sql_query_values(var, var, var, var): var
# build_sql_select(var, var): var
# build_sql_select_count(var, var): var
# build_sql_select_first(var): var
# build_sql_select_first_by_params(var): var
# build_sql_select_last(var): var
# build_sql_select_last_by_params(var): var
# build_sql_select_sum(var, var): var
+ count(): var
+ countByParams(var, var): var
+ create_link(var): var
+ create_search(var): var
+ criterion(): var
+ criterion_value(): var
+ delete(): var
+ destroy_link(): var
+ destroy_search(): var
# execute(var): var
# executeInsert(var): var
# executeUpdate(var): var
+ exist_advanced_search(): var
+ exist_criterion(): var
+ exist_criterion_value(): var
+ exist_link(): var
+ exist_search(): var
+ findAllFields(var, var): var
+ first(): var
+ firstByParams(var, var): var
+ get_data_system(): var
- get_field(var, var): var
+ get_fk(var, var): var
# get_properties_class(var): var
+ get_string_pks(var): var
+ get_string_values(var, var): var
+ get_sub_title(): var
# getCount(var): var
# getList(var, var): var
# getRow(var, var): var
# getSum(var): var
+ is_enabled(var, var): var
+ is_file(var, var): var
+ is_fk(var, var): var
+ is_fk_to(var, var): var
+ is_uni(var, var): var
+ last(): var
+ lastByParams(var, var): var
+ limit(): var
+ link(): var
+ load(var): var
+ loadByPk(var): var
+ page(): var
+ params(var): var
+ queryAll(var): var
+ queryByParams(var, var): var
+ queryBySqlExpression(var): var
# querySingleResult(var): var
# readCount(var): var
# readRow(var, var): var
# readSum(var): var
+ reset_sub_title(): var
+ save(): var
+ search(): var
+ set_sub_title(var): var
- setLastVarSqlCreate(var): var
- setLastVarSqlUpdate(var): var
+ sumByParams(var, var): var
+ to_string(): var
+ update(): var
Roles
+ _pk: var = 'id'
+ _pk_ai: var = true
- _set_conn_property: var = false
+ _singular: var = 'role'
# _table: var = 'roles'
+ descripcion: var
+ estado: var
+ estructura: var
+ id: var
+ rol: var
+ sistema: var
# connection_property(): var
+ define_file_fields(): var
+ define_fks(): var
+ define_fks_to(): var
+ define_unique_fields(): var
+ to_show(): var
Sistemas
+ _pk: var = 'id'
+ _pk_ai: var = true
- _set_conn_property: var = false
+ _singular: var = 'sistema'
# _table: var = 'sistemas'
+ descripcion: var
+ estado: var
+ id: var
+ sistema: var
# connection_property(): var
+ define_file_fields(): var
+ define_fks(): var
+ define_fks_to(): var
+ define_unique_fields(): var
+ to_show(): var
Campos
+ _pk: var = 'field'
+ _pk_ai: var = false
- _set_conn_property: var = false
+ _singular: var = 'role'
# _table: var = 'roles'
+ default: var
+ extra: var
+ field: var
+ key: var
+ null: var
+ type: var
# connection_property(): var
+ define_file_fields(): var
+ define_fks(): var
+ define_fks_to(): var
+ define_unique_fields(): var
+ getFields(var): var
+ to_show(): var
Usuarios
+ _pk: var = 'id'
+ _pk_ai: var = true
- _set_conn_property: var = false
+ _singular: var = 'usuario'
# _table: var = 'usuarios'
+ clave: var
+ descripcion: var
+ estado: var
+ id: var
+ usuario: var
# connection_property(): var
+ define_file_fields(): var
+ define_fks(): var
+ define_fks_to(): var
+ define_unique_fields(): var
+ to_show(): var
Relaciones
+ _pk: var = 'cOLUMNNAME'
+ _pk_ai: var = false
- _set_conn_property: var = false
+ _singular: var = ''
# _table: var = ''
+ column_name: var
+ constraint_schema: var
+ constraint_type: var
+ dis: var
+ referenced_column_name: var
+ referenced_table_name: var
+ referenced_table_schema: var
+ table_name: var
# connection_property(): var
+ define_file_fields(): var
+ define_fks(): var
+ define_fks_to(): var
+ define_unique_fields(): var
+ getIndexMultiUnique(var, var): var
+ getRelatedDB(var): var
+ getRelations(var, var, var): var
+ getRelationsreferences(var, var, var): var
+ to_show(): var
Tablas
+ _pk: var = 'tablesIn'
+ _pk_ai: var = false
- _set_conn_property: var = false
+ _singular: var = 'tabla'
# _table: var = ''
+ tablesIn: var
+ createDBInit(var): var
+ createTableManuales(var): var
+ createTableRoles(var): var
+ createTableRolesUsuarios(var): var
+ createTableSistemas(var): var
+ createTableUsuarios(var): var
- getListTables(var): var
+ getTables(): var
- readRowTable(var): var

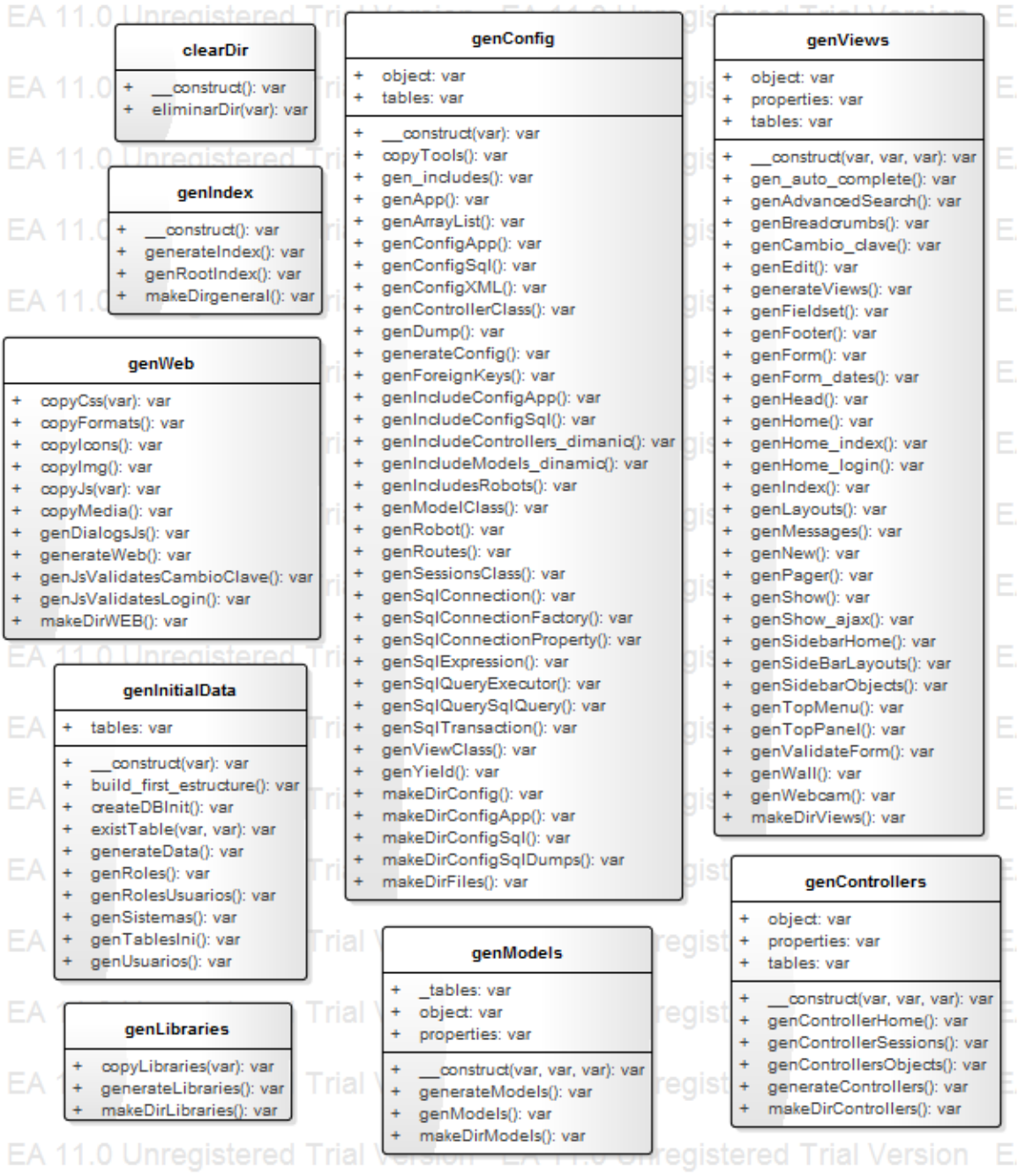


Ilustración 24. Diagrama de Clases

El Diagrama de Paquetes se creó con la herramienta StartUML, citada anteriormente, donde se identificaron seis paquetes; vistas, controladores, modelos, config, generator y librerías.

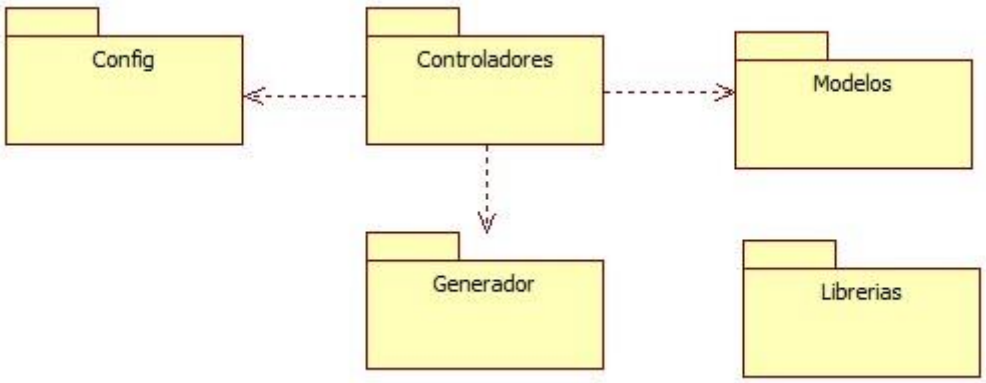


Ilustración 25. Diagrama de paquetes

Las clases y/o sub paquetes que confirman cada paquete son:

Paquete	**Clases**
Config	• Routes • Sessions • Tolos • Tolos_tgenp • Connection • ConnectionFactory • ConnectionProperty • Foreignkeys • QueryExecutor • SqlExpression • SqlQuery • Transaction
controladores	• All_controller • Clear_controller • Config_controller • Controllers_controller • Index_controller • Initial_data_controller • Libraries_controller • Models_controller • Views_controller • Web_controller

Modelos	• Campos • Model • Relaciones • Roles • Roles_usuarios • Sistemas • Tablas • Usuarios
Generador	• clearDir • genConfig • genControllers • genIndex • genInitialData • genLibraries • genModels • genViews • genWeb
Librerías	• PHPExcel (Paquete) • PHPMailer (Paquete) • Pclzip (Paquete) • pdfClassesAndFonts (Paquete)

En el paquete “Libraries” se listan solo los sub paquetes que lo conforman, ya que cada sub paquete contiene un número muy alto clases y no hacen parte de las clases de misión crítica del sistema, debido a lo cual tampoco que incluyeron en el Diagrama de Clases.

Análisis de código fuente

En base a los factores que definieron y delimitaron los componentes a tener en cuenta en la generación de los diagramas UML explicados en el inicio del presente apartado, se crearon dos Diagramas de actividades: Diagrama de Actividades De Conectar a Base de Datos (Ilustración 26) y Diagrama de Actividades de Generar proyecto Completo (Ilustración 27), dos diagramas de Secuencias: Diagrama de secuencias Conectar a Base de Datos (Ilustración 28) y Diagrama de secuencias Generar proyecto Completo (Ilustración 29), y el diagrama de componentes (Ilustración 30).

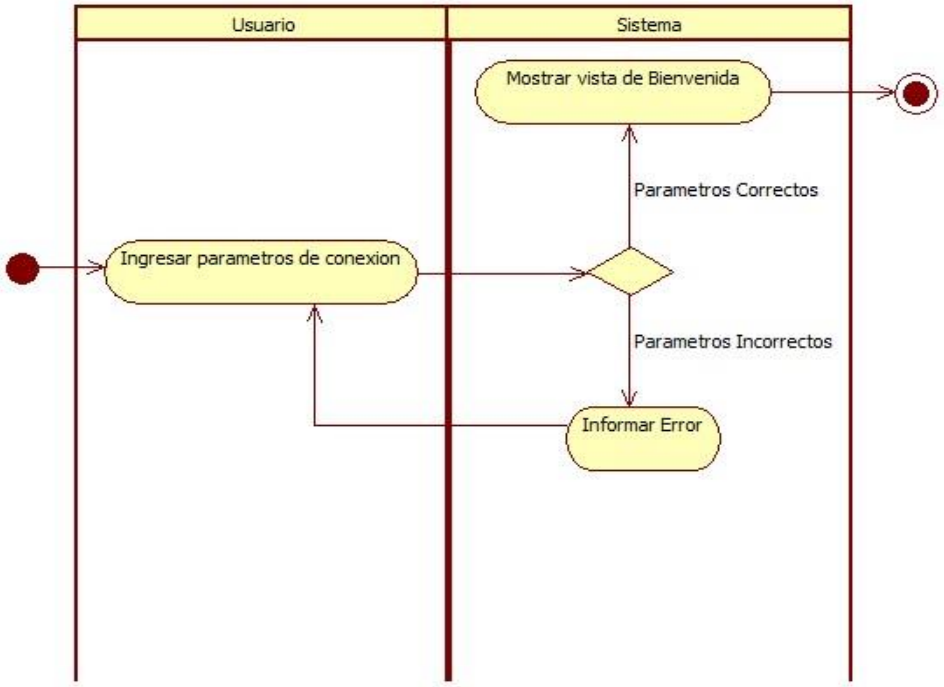


Ilustración 26. Diagrama de Actividad Conectar a Base Datos

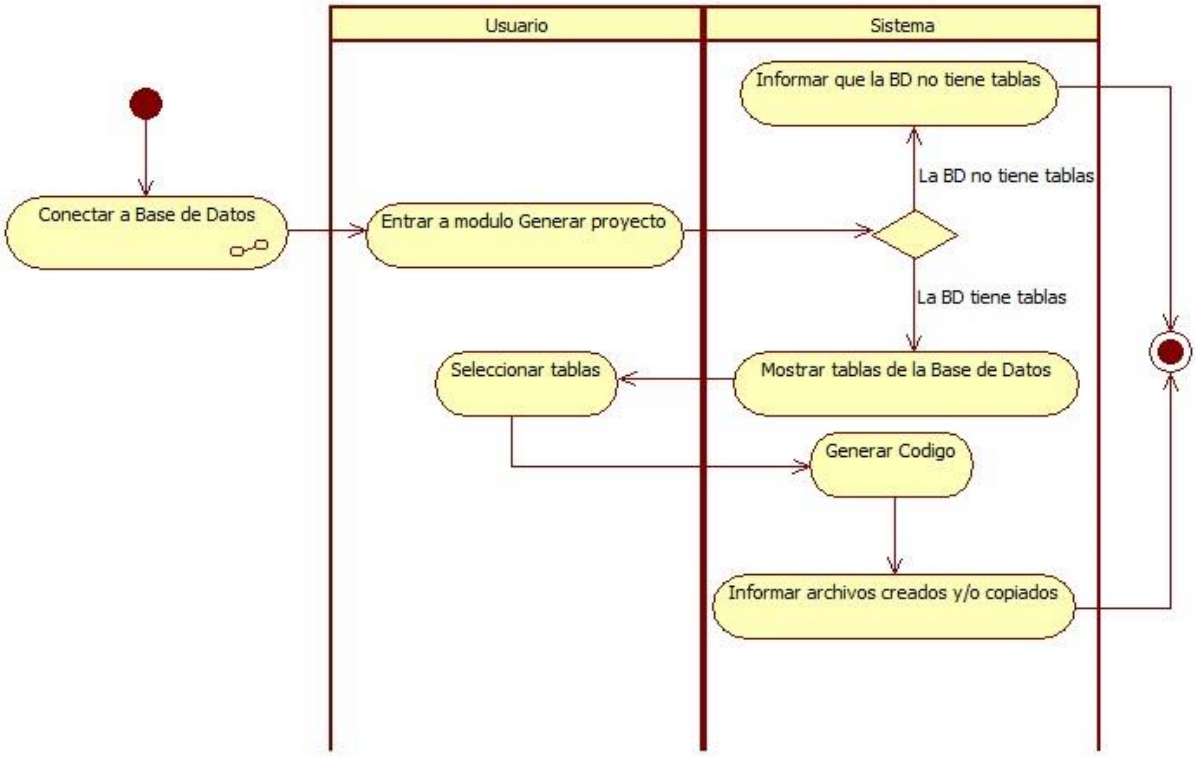


Ilustración 27. Diagrama de Actividad Generar Proyecto

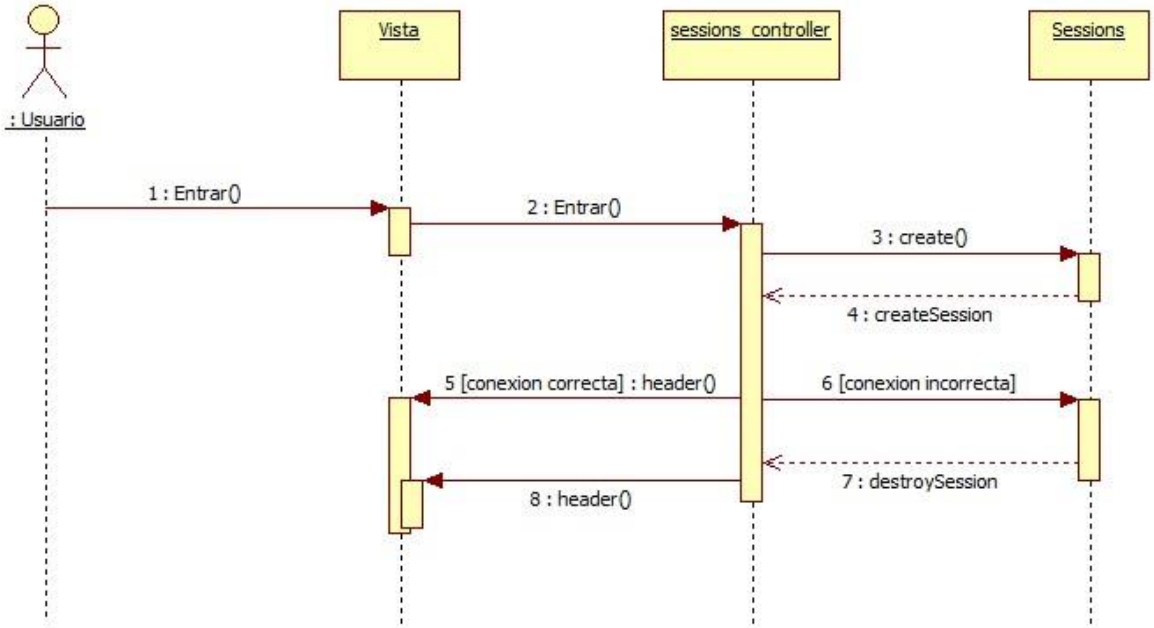


Ilustración 28. Diagrama de Secuencia Conectar Con Base de Datos

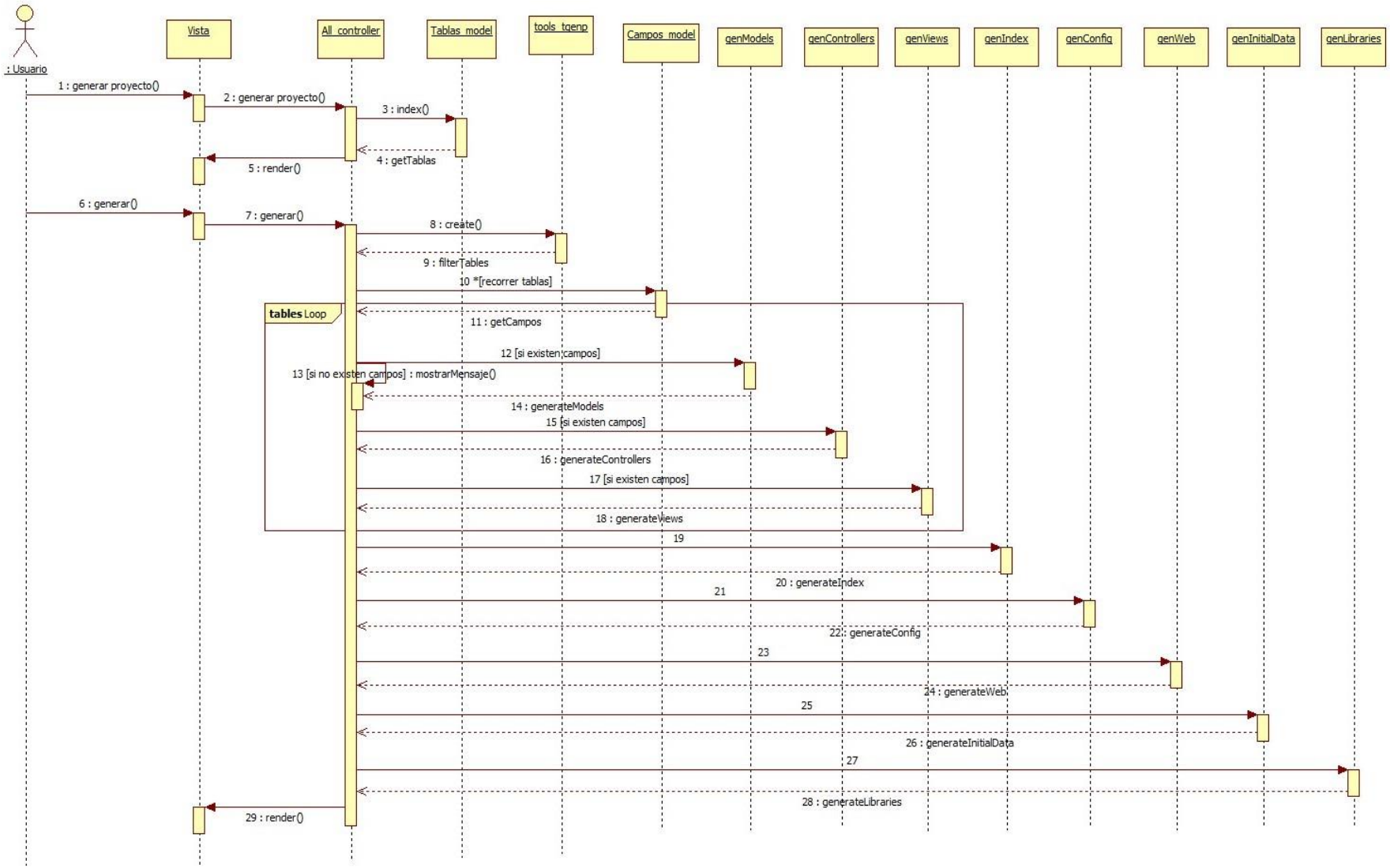

: Usuario
Vista
All_controller
Tablas_model
tools_tgenp
Campos_model
genModels
genControllers
genViews
genIndex
genConfig
genWeb
genInitialData
genLibraries
1 : generar proyecto()
2 : generar proyecto()
3 : index()
4 : getTablas
5 : render()
6 : generar()
7 : generar()
8 : create()
9 : filterTables
10 *[recorrer tablas]
tables Loop
11 : getCampos
12 [si existen campos]
13 [si no existen campos] : mostrarMensaje()
14 : generateModels
15 [si existen campos]
16 : generateControllers
17 [si existen campos]
18 : generateViews
19
20 : generateIndex
21
22 : generateConfig
23
24 : generateWeb
25
26 : generateInitialData
27
28 : generateLibraries
29 : render()

Ilustración 29. Diagrama de secuencias Generar proyecto Completo

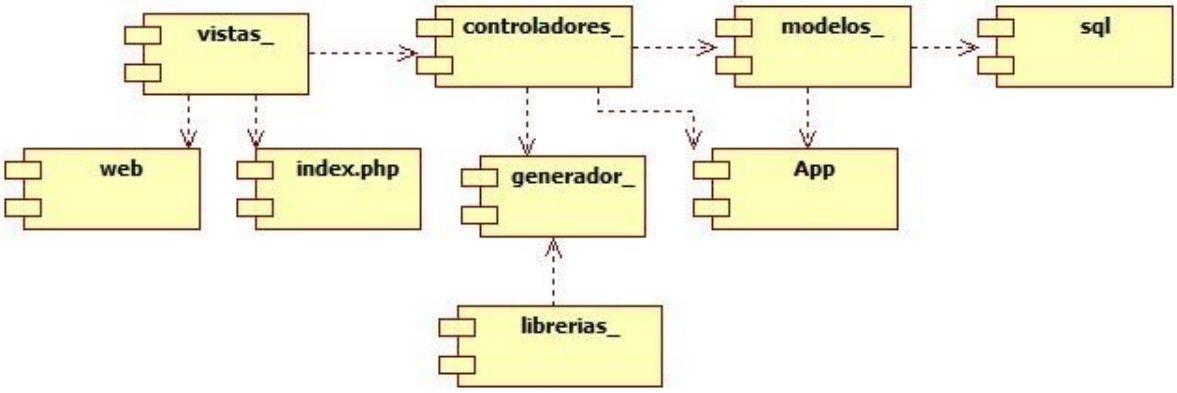


Ilustración 30. Diagrama de Componentes

REFERENCIAS

- [1] Frameworks de desarrollo: un método ágil para el desarrollo de software. 17 de noviembre de 2010. http://www.opiniontecnologica.com/desarrollo-web/frameworks-de-desarrollo-un-metodo-agil-para-el-desarrollo-de-software.html. **Eduardo Morcillo**

- [2] Lenguaje Unificado de Modelado (UML). http://www.disca.upv.es/enheror/pdf/ActaUML.PDF. Enrique Hernández Orallo.

- [3] ¿Qué es la Ingeniería Inversa? http://cnx.org/content/m17432/latest/. Miguel-Angel Sicilia

- [4] Uso de la notación UML en el desarrollo de aplicaciones educativas. 2000. Antonio Edwin Benavente Morales, Universidad Nacional de San Agustín, Núcleo de NTIC en la Educación, Perú

- Trabajo de Grado Presentado como Requisito Parcial para Optar al Título de Ingeniero De Sistemas "Diseño De Un Sistema De Información Para La Gerencia De Ventas De Una Empresa De Mantenimiento Y Suministro De Equipos Analíticos De Laboratorio, Ubicada En Puerto Ordaz, Estado Bolívar". Romanelli M., Romina C., López A. María C.

- CommonKADS y el Lenguaje de Modelado Unificado .UML, Pedro Salcedo Lagos, Departamento de Metodología de la Investigación e Informática Educativa, Facultad de Educación, Universidad de Concepción CHILE - psalcedo@udec.cl

- GISWEB, Reingeniería para la creación de un WEB Feature Service. Recuperado el 17 de junio de 2014. http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/lopez_a_aa/capitulo4.pdf.

- Enciclopedia libre. Recuperado el 17 de junio de 2014. http://es.wikipedia.org/wiki/Modelo_Vista_Controlador

- UML: Diagramas UML. ¿Qué es UML? 2005. http://www.ingenierosoftware.com/analisisydiseno/uml.php. Joaquín Gracia

- Aprendiendo UML en 24 horas. Joseph Schmuller. Prentice Hall.

- Porque es importante UML? publicado el 31/12/2007 a las 22:15. http://www.osmosislatina.com/lenguajes/uml/basico.htm.

- Ingeniería Inversa y Reingeniería Aplicadas a Proyectos de Software Desarrollados por Alumnos de Nivel Licenciatura. 2007. http://www.iiisci.org/journal/CV$/risci/pdfs/X581YP.pdf. Reyes Juárez Ramírez, Guillermo Licea, Alfredo Cristóbal Salas

- Ingeniería inversa aplicada a sistemas desarrollados con programación orientada a objetos para obtener la documentación. Universidad Nacional Mayor de San Marcos. 2007. http://cybertesis.unmsm.edu.pe/bitstream/cybertesis/2626/2/acevedo_rj.pdf. Jessica Jahany Acevedo Ricse, Elmer Emilio Puma Falcón